Move ordering in chess using realization probability search

Dirk Wuytack
851353247
20 april 2017

Master thesis

Open Universiteit
www.ou.nl

Move ordering in chess using realization probability search

Master thesis

Open Universiteit, faculteit Management, Science & Technology

Masteropleiding Software Engineering

Student:	Dirk Wuytack
Student Number:	851353247
Course Number:	T75317
Date:	20/04/2017
Chairman:	Prof.dr. M.C.J.D. van Eekelen
Supervisor:	dr. A.J. Hommersom

Summary

Thanks to ever-faster processors and advanced algorithms, chess programmes have already become significantly stronger than their human opponents (1). But the complexity of chess (see 1.4 complexity of a chess game) is of such an order that - theoretically - it is still impossible to calculate all the variants in a chess position to find the best move.

The classic approach for making chess programmes more performant is to selectively choose the moves. This is, in fact, what the human brain is also very good at. Indeed, a chess grandmaster overlooks a chess position and is able to tell – in just a few seconds - which moves are worth considering. Modern chess programmes simulate this behaviour by means of a move ordering algorithm (see 2.2 Move ordering algorithms). This algorithm sorts the available moves from best to worst by means of a forecast. This will enable the alpha-beta pruning algorithm (2.4 Alpha-beta pruning algorithm) to remove a large number of variants without risking to overlook the best move in doing so. The better the move ordering, the more variants can be discarded and the deeper a chess program will calculate.

Realization probability search as a move ordering algorithm has already been successfully tested in other thinking games (2) (3) but not yet in chess programmes. In this research we have tested the usability of RPS[1] as a move ordering algorithm in a chess programme. RPS[1] predicts the best variant by means of statistical data, using a probability table that contains the play probabilities of possible move categories. The probability that a variant will be played in a chess position is the product of the probabilities of the move categories which the moves of that variant belong to.

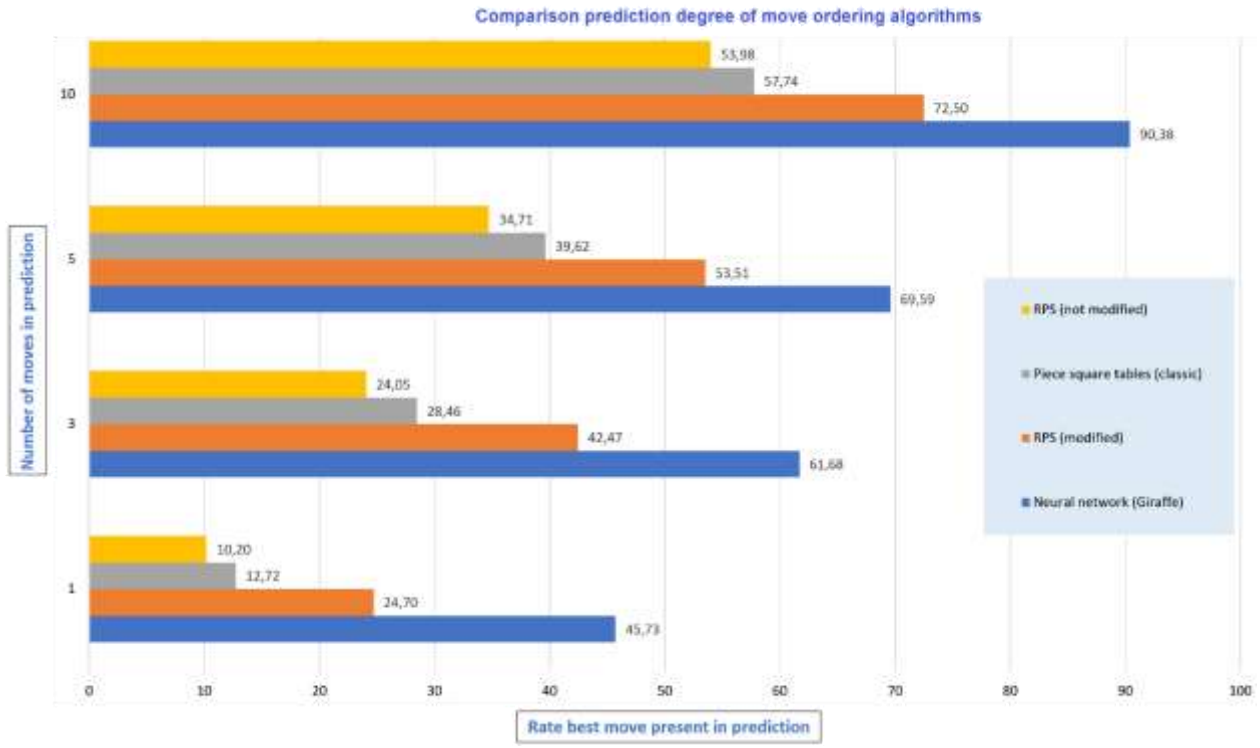
To generate this probability table we have analysed 3,000 top level chess games with self-written software. 124 move categories, compiled by studying chess literature, were checked. We've implemented and evaluated RPS[1] using two important criteria: the working speed and the predictive power. Due to the lack of a gold standard we have compared RPS[1] with an own implementation of a classic move algorithm (see 6. Piece square tables). The results were also compared with the move ordering algorithm used in Giraffe (4). Giraffe is a chess program that plays at the level of international master (approximately 2,400 ELO points[2]) and uses a neural network[3] to implement the move ordering.

Based on the results obtained we can conclude that move ordering through RPS[1] in itself has less predictive power than a classic move ordering algorithm. If, however, we break down the move categories into four groups (captures, favourable moves, normal moves and bad moves) and give the groups priority over one another, the results suddenly improve by far. Within those four groups each category maintains its own probability index as they appear in the probability table.

Figure 1 shows the comparison between the implemented move algorithms together with the results of the move ordering algorithm used in Giraffe (4). 1,874 positions were examined in twenty games which were played between chess programmes with a rating of at least 3,000 ELO[2] points. In a chess position 32 moves are possible on average. RPS[1] (modified) is able to predict the best move in almost

[1] Realization probability search.
[2] ELO Points: reflect the strength of a player. The strongest human player has 2,853 points and the strongest chess program 3,393 points.
[3] Neural network: a network specialized in recognizing relationships and patterns

25% of all test positions and can place the best move in 42% of the top three. In 53% of all cases RPS[1] puts the best move in the top five and in 72% of all cases in the top ten(see figure 1).



Figuur 1: Average prediction power of RPS[1] (twenty testmatches - 1874 chesspositions)

The modified version of RPS[1] is clearly better than a classic move ordering algorithm (see 6. Classic move ordering) but it is less accurate than a move ordering based on a neural network (4).

However all used move ordering algorithms operate at a working speed of less than a second on an normal desktop pc (which is an extremely important criteria) except the one based on a neural network. The speed of the neural network used in Giraffe (4) is - according to the author - a negative aspect. The neural network is so slow (24) that the move ordering algorithm - even on a very powerful machine – not always succeed to make a prediction and consequently is not suitable as a move ordering system for chess programmes that run on a normal desktop pc.

We have in this research RPS[1] compared with only one classic move ordering algorithm. Based on this research, we can conclude that RPS[1] is probably useful as a move ordering system in a chess programme. However many chess programmes make use of a combination of move ordering techniques. Future studies will have to point out whether RPS[1] in combination with another complementary move ordering algorithm can still push up its predictive power.

[1] Realization probability search.

Samenvatting

Dankzij steeds snellere processoren en geavanceerde algoritmes spelen schaakprogramma's al beduidend sterker dan hun menselijke tegenstanders (1). Maar de complexiteit van het schaken (zie 1.4 De complexiteit van het schaakspel) is van zulke orde dat een schaakprogramma nog steeds onmogelijk alle varianten in een schaakpositie volledig door kan rekenen om zodoende tot de beste zet te komen.

Een klassieke manier om een schaakprogramma zo diep mogelijk te laten rekenen, is selectief te werk gaan. Dit is in feite waar het menselijk brein ook zeer goed in is. Een schaakgrootmeester overziet een stelling en weet na luttele seconden welke zetten het overwegen waard zijn. Moderne schaakprogramma's bootsen dit gedrag na door middel van een move ordering algoritme (zie 2.2 Move ordering algoritme). Dit algoritme sorteert de beschikbare zetten in een schaakpositie van beste naar slechtste zet op basis van een prognose. Door middel van het alpha-beta pruning algoritme (zie 2.4 Alpha-beta pruning algoritme) kan dan een groot aantal varianten worden geschrapt zonder het risico de beste zet over het hoofd te zien. Hoe beter de move ordering, hoe meer varianten er kunnen worden geschrapt tijdens het evalueren van de beschikbare zetten en hoe dieper een schaakprogramma kan rekenen.

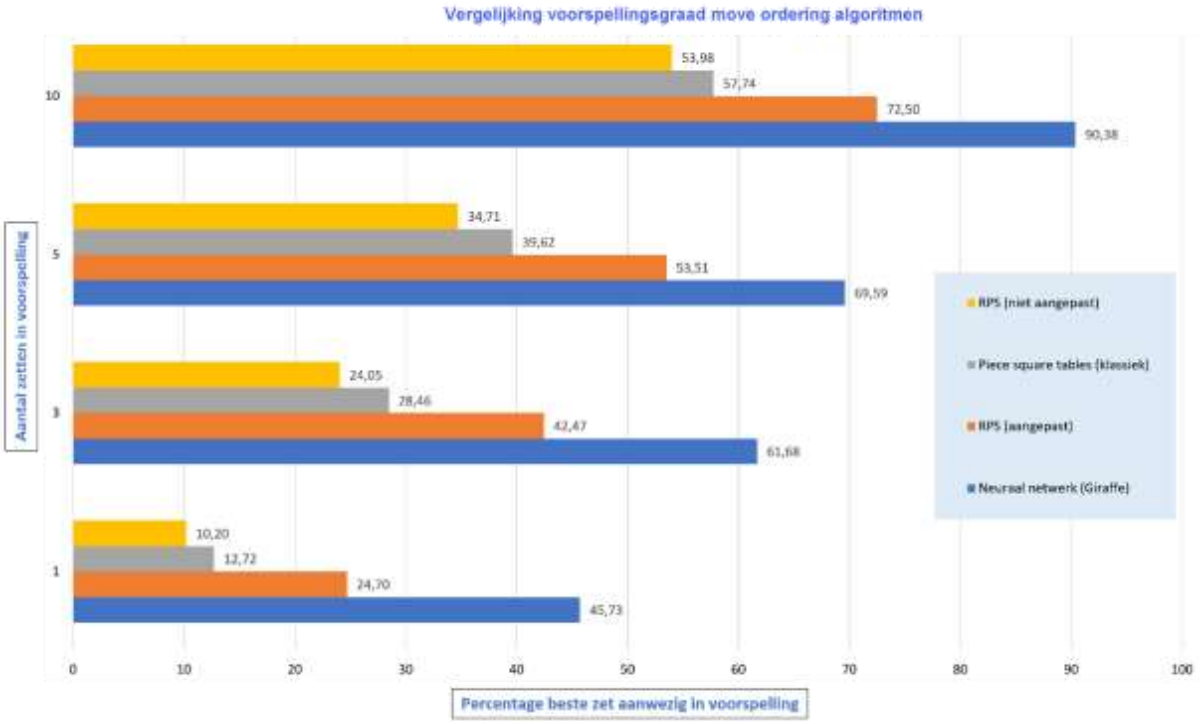
Realization probability search (RPS) als een move ordering techniek is al succesvol getest in andere denkspelen(2)(3), maar nog niet in een schaakprogramma. RPS[1] bepaalt de move ordering door gebruik te maken van een probability tabel die de speelkansen bevat van mogelijke zetcategorieën. De kans dat een variant wordt gespeeld in een bepaalde positie, wordt bepaald door het product van de probability indexen van de zetcategorieën waartoe de zetten van die variant behoren. We hebben in dit onderzoek de bruikbaarheid van RPS[1] getest in een schaakprogramma.

De probability tabel hebben we gegenereerd door meer dan 3.000 partijen op topniveau te analyseren. 124 zetcategorieën, die we hebben samengesteld aan de hand van informatie uit de schaakliteratuur, werden daarbij gecontroleerd. We hebben RPS[1] geïmplementeerd en geëvalueerd aan de hand van twee belangrijke criteria: de voorspellingsgraad en de snelheid. Door het gebrek aan een gouden standaard hebben we RPS[1] vergeleken met een eigen implementatie van een klassiek move ordering algoritme (zie 6. Klassieke move ordering (piece square tables)). De resultaten werden ook vergeleken met de move ordering techniek die werd gebruikt in Giraffe(4). Dit is een schaakprogramma dat speelt op internationaal meester niveau (ongeveer 2.400 ELO-punten[2]) en gebruik maakt van een neuraal netwerk[3] voor de move ordering.

Uit de bekomen resultaten kunnen we besluiten dat move ordering door middel van RPS[1] zonder enige aanpassingen minder voorspellende kracht heeft dan een klassiek move ordering algoritme. Als we echter de zetcategorieën opsplitsen in vier groepen (captures, gunstige zetten, normale zetten en slechte zetten) en we verlenen deze groepen voorrang op elkaar, dan worden de bekomen resultaten plots een pak beter. In figuur 2 is de voorspellingsgraad van de geïmplementeerde move ordering technieken vergeleken met de resultaten van de move ordering techniek die gebruikt is in het

[1] Realization probability search
[2] ELO-punten: geeft de sterkte van een speler weer. De sterkste menselijke speler heeft 2853 punten en het sterkste schaakprogramma 3393 punten
[3] Een rekennetwerk dat wordt toegepast voor het vinden van relaties en het herkennen van patronen in datasets

schaakprogramma Giraffe (4). Er werden in totaal 1.874 schaakposities onderzocht uit twintig testmatchen die gespeeld zijn tussen schaakprogramma's van top niveau met een rating van minstens 3.000 ELO[1] punten. Gemiddeld is er in een schaakpositie keuze uit 32 zetten. De aangepaste variant van RPS[2] kan in gemiddeld 25% van alle testposities de beste zet voorspellen, in 42% van alle testposities zit de beste zet in de top drie, in 53% in de top vijf en 72% in de top tien.



Figuur 2: Gemiddelde voorspellingsgraad van RPS[2] (twintig testmatchen - 1874 schaakposities)

RPS[2] (aangepaste variant) scoort beduidend beter dan een klassieke move ordering techniek (zie 6. Piece square tables), maar minder accuraat dan een move ordering techniek die gebruik maakt van een neuraal netwerk (4).Alle gebruikte move ordering methoden werken echter aan een snelheid van minder dan één seconde (een zeer belangrijk criteria), behalve de techniek die is gebaseerd op een neuraal netwerk. De snelheid van het neuraal netwerk, dat wordt gebruikt in Giraffe(4), is volgens de auteur (Matthew Lai) een negatief aspect. Het maakt de move ordering dermate traag (24) dat het zelfs op zeer krachtige machines er soms niet in slaagt een voorspelling te doen binnen de toegestane tijd en bijgevolg niet bruikbaar is als move ordering systeem voor schaakprogramma's die draaien op een normale desktop PC. We hebben RPS[2] met maar één klassiek move ordering systeem vergeleken.

Uit deze context kunnen we concluderen dat RPS[2] waarschijnlijk goed bruikbaar is als move ordering systeem in een schaakprogramma. Veel schaakprogramma's gebruiken echter een combinatie van move ordering technieken om de resultaten te verbeteren. Toekomstige onderzoeken zullen moeten uitwijzen of RPS[2] in combinatie met andere move ordering technieken zijn voorspellend vermogen nog kan opdrijven.

[1] [1] ELO-punten: geeft de sterkte van een speler weer. De sterkste menselijke speler heeft 2853 punten en het sterkste schaakprogramma 3393 punten.

[2] RPS: Realization probability search.

Inhoudsopgave

Summary 3
Samenvatting 5
Inhoudsopgave figuren 9
1. **Inleiding** 10
1.1 Kunstmatige intelligentie in denkspelen 10
1.2 Brute force techniek versus intelligente algoritmen 10
1.3 Kunstmatige intelligentie in een schaakprogramma 11
1.4 Complexiteit van het schaakspel 11
2. **Achtergrond en literatuur** 13
2.1 Move list algoritme 14
2.2 Move ordering algoritmen 14
2.2.1 History heuristic 15
2.2.2 Static exchange evaluation 15
2.2.3 Piece-square tables 16
2.2.4 Iterative deepening 16
2.2.5 Move influence 17
2.2.6 Bayesian pattern ranking 17
2.2.7 Monte Carlo algoritme 18
2.2.8 Transposition tables 18
2.2.9 Realization Probability Search 18
2.3 Evaluatiefunctie 19
2.4 Alpha-beta pruning algoritme 19
2.5 Search algoritme 22
2.6 Game list algoritme 24
3. **Onderzoeksmodel** 25
4. **Probability tabel** 27
4.1 Zetcategorieën 27
4.2 Probability index 27
5. **Realization probability search** 29
5.1 Achtergrond 29
5.2 Algoritme 31
6. **Klassieke move ordering (piece square tables)** 33
7. **Resultaten** 34
7.1 Realization probability search 34

7.2 Realization probability search (aangepast) 34
7.2 Klassieke move ordering variant (piece square tables) 36
7.3 Neuraal netwerk 36
8. **Conclusie** 38
8.1 Discussie 38
8.2 Toekomstig werk 39
Literatuurlijst 40
Bijlage 1: Afkortingen 42
Bijlage 2: Verklarende woordenlijst 43
Bijlage 3: Zet categorieën. 45
Bijlage 4: Handleiding software 51
analyseCategorienDetail.jar 51
analyseCategorienSamenvatting.jar 55
RPS.jar / RPS_nietAangepast.jar 58
KlassiekMoveOrdering.jar 62
Bijlage 5: Probability Tabel 66
Bijlage 6: Piece-square tables 68
Bijlage 7: Resultaten voor alle geïmplementeerde move ordering algoritmen. 69

Inhoudsopgave figuren

Figuur 1: Average prediction power of RPS[1] (twenty testmatches - 1874 chesspositions) 4
Figuur 2: Gemiddelde voorspellingsgraad van RPS[2] (twintig testmatchen - 1874 schaakposities) 6
Figuur 3: Een overzicht van de move algoritmen in een schaakprogramma 13
Figuur 4: Een voorbeeld van een zeer eenvoudige zoekboom 14
Figuur 5: Enkele verschillende move ordering technieken 15
Figuur 6: Een mogelijke piece-square table voor de witte pionnen 16
Figuur 7: Iterative deepening: het zoeken laag per laag 17
Figuur 8: Een voorbeeld hoe het bord kan worden ingedeeld in zones 17
Figuur 9: Vierstappen plan van het Monte Carlo algoritme (8) 18
Figuur 10: De eerste alpha waarde bepalen 20
Figuur 11: De tweede groep levert een nieuwe alpha waarde op 20
Figuur 12: Beperking zoekboom door het alpha beta pruning algoritme 21
Figuur 13: Als de move ordering er in slaagt om de beste zet eerst te zetten, kan er maximaal worden gesnoeid! 22
Figuur 14: Het minimax algoritme toepassen 23
Figuur 15: De resultaten van de evaluatiefunctie 23
Figuur 16: Het bepalen van de hoofdvariant 24
Figuur 17: Gedeelte van de gegenereerde probability tabel 28
Figuur 18: Het berekenen van de probability van een bepaalde positie 29
Figuur 19: Testresultaten van (E)RPS 30
Figuur 20: Gemiddelde voorspellingsgraad: 20 testmatchen (1.874 plies) 36

1. Inleiding

1.1 Kunstmatige intelligentie in denkspelen

Wat is kunstmatige intelligentie? In de loop der tijd zijn verschillende definities geformuleerd, maar volgens het gerenommeerde naslagwerk “Artificial Intelligence” (5) luidt de definitie: “een machine bezit kunstmatige intelligentie als het rationeel kan functioneren en denken als een mens”.

Kunstmatige intelligentie heeft mij altijd gefascineerd en in het bijzonder in schaakprogramma’s, vooral omdat ikzelf een (bescheiden) clubspeler ben. Een van de grootste mijlpalen op het gebied van kunstmatige intelligentie in denkspelen was in het spel Go: op 15 maart 2016 versloeg AlphaGo, dat werd ontwikkeld door Google, een speler uit de top vijf (Lee Sedol) in een vijf partijen durende match (4-1) (6). Dit was vooral indrukwekkend omdat in Go een speler gemiddeld kan kiezen uit een indrukwekkende 250 zetten. AlphaGo gebruikt onder andere een database van 30 miljoen zetten uit historische partijen van professionele spelers om beter te leren spelen.

Er werden ook succesvolle resultaten geboekt in het denkspel Shogi door gebruik te maken van statistische data uit een database van 600 partijen. Bij Shogi tekende men 72% winst op ten opzichte van de klassieke move ordering technieken (zie 2.2 Move ordering algoritmen). Kunnen we in het schaken de expertise die in eerder gespeelde partijen op top niveau zit ook niet hergebruiken om een schaakprogramma sterker te laten spelen?

1.2 Brute force techniek versus intelligente algoritmen

De eerste denkspelen op de computer gebruikten alleen de brute force techniek en rekenden dus gewoon zoveel mogelijk posities uit. De enige vorm van intelligentie was het gebruik van een beslissingsalgoritme (dat de sterkste variant kiest) dat tot op de dag van vandaag nog altijd wordt gebruikt: het minimax algoritme dat ontworpen is door C. Shannon in 1950 (7) (zie 2.5 Search algoritme).

De brute force techniek bleek niet voldoende om de topspelers in denkspelen te verslaan omdat er op een gegeven ogenblik gewoonweg te veel spelvarianten ontstaan (zie 1.4 Complexiteit van het schaakspel). Men kon al deze spelvarianten niet door rekenen, zelfs niet met de huidige krachtige processoren. Er moest een oplossing worden gevonden om het gigantisch aantal nodes[1] in een spelpositie terug te dringen naar een doorzoekbaar geheel. Al heel snel werd het alpha-beta pruning algoritme (zie 2.4 Alpha-beta pruning algoritme) ontwikkeld dat het aantal te doorzoeken varianten sterk kan terugdringen zonder het risico de beste variant over te slaan. De effectiviteit waarmee het alpha beta pruning algoritme werkt, hangt echter sterk af van de volgorde van de varianten die worden onderzocht.

Het alpha-beta pruning algoritme kan een maximaal aantal nodes snoeien, als de beste zetten eerst worden behandeld (zie 2.4 Alpha-beta pruning algoritme). Zonder alle varianten te evalueren, weet men natuurlijk niet wat de beste beschikbare varianten zijn. Daarom heeft men een algoritme ontwikkeld dat alle beschikbare zetten analyseert en op basis daarvan een prognose maakt en de zetten sorteert van beste zet naar slechtste zet. De analyse die dit algoritme uitvoert moet aan twee

[1] Node: knoop in een zoekboom

voorwaarden voldoen: Het moet zo accuraat en zo snel mogelijk zijn. Dit algoritme noemt men het move ordering algoritme. Men heeft verschillende soorten move ordering algoritmen afhankelijk van het soort techniek dat wordt gebruikt om de beschikbare varianten te analyseren. In hoofdstuk 2.2 vindt u een gedetailleerde beschrijving van de verschillende move ordering technieken.

Hoe beter de sortering gebeurt door het move ordering algoritme, hoe meer varianten het alpha-beta algoritme kan snoeien. De overblijvende varianten worden dan aan de evaluatiefunctie doorgegeven die dan een doorgedreven analyse uitvoert om de beste zet te bepalen. Het mooie aan het alpha-beta pruning algoritme is dat het varianten kan snoeien zonder het risico dat de beste variant bij de gesnoeide varianten zit. Een gedetailleerde beschrijving van het alpha-beta pruning algoritme vindt u terug in hoofdstuk 2.4.

1.3 Kunstmatige intelligentie in een schaakprogramma

Al deze verbeteringen hebben ervoor gezorgd dat de huidige schaakprogramma's veel sterker spelen dan de huidige wereldkampioen Magnus Carlsen. Heeft het dan nog zin om de schaakprogramma's nog verder te verbeteren? Het streefdoel is het schaakspel volledig theoretisch door te rekenen zodat bij elke positie werkelijk de beste zet kan worden gekozen. Een schaakprogramma kan dan als perfecte trainer dienen voor de menselijke spelers. Elke verbetering, hoe klein ook, brengt ons dichter bij dat doel. Bij uitbreiding kunnen dan de algoritmes ook gebruikt worden in andere software om deze dan ook efficiënter te maken.

Het verschil in sterkte tussen de beste schaakprogramma's is vooral terug te brengen tot de manier waarop zij de besproken algoritmen implementeren. Bij de move ordering algoritmen kan bijvoorbeeld gekozen worden uit verschillende technieken (zie 2.2 Move ordering algoritmen).

In dit onderzoek is een nieuwe techniek getest als move ordering algoritme in een schaakprogramma: Realization Probability Search. Deze techniek gebruikt statistische data om de mogelijke zetten te sorteren van best naar slechtst. In ons onderzoek hebben we die data verkregen uit een database met meer dan 3.300 top games tussen schaakprogramma's die minstens 3.000 ELO[1]-punten hebben. Ter vergelijking: de huidige menselijke wereldkampioen Magnus Carlsen voert de wereldranglijst aan met 2.838 ELO[1] punten (8). Hoe realization probability search precies werkt, wordt uitvoerig in hoofdstuk vijf beschreven.

1.4 Complexiteit van het schaakspel

In het schaakspel zijn er 10^{120} schaakposities mogelijk waarvan 10^{43} schaakposities legaal zijn. We hebben een move generator geïmplementeerd die alle mogelijke varianten genereert vanaf de beginstelling van het schaakspel en enkel de unieke posities bewaart. Het resultaat ziet u in tabel 1.

Het aantal mogelijke legale posities na zeven plies[2] bedraagt meer dan drie miljard posities. Een schaakprogramma op een moderne desktop pc kan brute force alle mogelijke varianten uitrekenen tot

[1] ELO-punten: geeft de sterkte van een speler weer. De sterkste menselijke speler heeft 2853 punten en het sterkste schaakprogramma 3393 punten

[2] Ply: één zet van één speler

maximum vijf plies[1] diep in een normale middenspelsituatie (binnen de toegestane tijd van twee minuten).

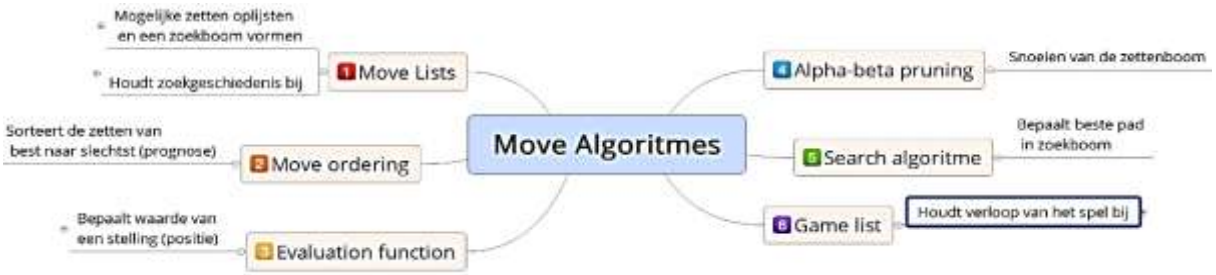
Softwarematig gaat men echter de gigantische zettenboom snoeien via het alpha-beta pruning algoritme (zie 2.4 Alpha-beta pruning algoritme), zodat de huidige schaakprogramma's soms tot 26 plies[1] diep rekenen. De effectiviteit van het alpha-beta pruning algoritme hangt echter in grote mate af van hoe accuraat de beschikbare zetten zijn gerangschikt van best naar slechtst. Het is dus enorm belangrijk dat het move ordering algoritme een zo juist mogelijke prognose maakt, zodat zonder verlies aan informatie het aantal te doorzoeken nodes enorm kan worden teruggedrongen.

Tabel 1: aantal posities mogelijk in het schaakspel vanaf de startpositie

Aantal plies[1] vanuit beginpositie	Aantal mogelijke posities	Aantal mogelijke unieke posities
1	20	20
2	400	400
3	8.902	5.362
4	197.281	72.060
5	4.865.609	822.199
6	119.060.324	9.396.023
7	3.195.901.860	96.400.068

2. Achtergrond en literatuur

Ondanks de krachtige hardware en gesofisticeerde software kan een huidig schaakprogramma nog altijd het schaakspel niet theoretisch perfect uit spelen. Kunnen de huidige schaakprogramma's dan nog verbeteren? De wet van Moore die stelt dat het aantal transistors op een processor om de twee jaar verdubbelt en dus de processor ook veel krachtiger wordt, gaat de laatste jaren niet meer op. De snelheid van de hardware zorgt dus steeds minder en minder voor prestatiewinst. Een andere manier om de schaakprogramma's sterker te maken, is de schaakalgoritmes te verbeteren of te vervangen. De meest gebruikte schaakalgoritmes op dit ogenblik zijn (zie figuur 4):



Figuur 3: Een overzicht van de move algoritmen in een schaakprogramma

De volgorde waarin deze move algoritmen worden uitgevoerd, is als volgt:

1. Move list genereren (2.1 Move list algoritme).

2. Move ordering.
   Move ordering sorteert de beschikbare legale zetten van best naar slechtst. Dit ter voorbereiding van het alpha-beta pruning algoritme die dan maximaal kan snoeien in de zettenboom. (2.2 Move ordering algoritmen)

3. Evaluatiefunctie.
   Evalueren van de verschillende varianten van deze zetten in volgorde van de gesorteerde lijst. De evaluatiefunctie is verantwoordelijk voor het bepalen van de waarde van een bepaalde spelpositie. (2.3 Evaluatiefunctie)

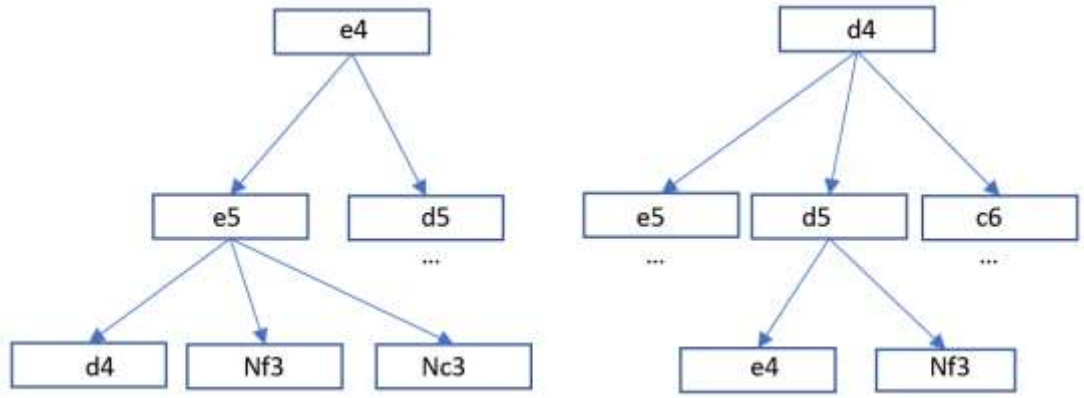
4. Alpha-beta pruning algoritme.
   Tijdens het evalueren wordt het alpha-beta pruning algoritme gebruikt om het aantal te evalueren stellingen te beperken. Hoe beter het move ordering algoritme zijn werk doet, hoe meer dat het alpha-beta pruning algoritme kan snoeien (2.4 Alpha-beta pruning algoritme).

5. Search algoritme.
   Bepalen van het beste pad of hoofdvariant (principal variation). Het zoekalgoritme staat in voor het bepalen van de beste variant die er kan worden gespeeld. (2.5 Search algoritme).

6. Game list algoritme.
   Lijst bijhouden van de gespeelde zetten (2.6 Game list algoritme).

2.1 Move list algoritme

Het move list algoritme genereert een zoekboom waarin alle mogelijke legale zetten en tegenzetten tot op een gewenste diepte zijn opgenomen. In onderstaande figuur ziet u een hypothetisch voorbeeld hoe een zeer eenvoudige zoekboom eruit ziet. Wit heeft keuze uit 2 zetten (e4 en d4). Indien hij bijvoorbeeld e4 kiest, dan kan zwart kiezen uit twee zetten (e5 en d5). Als zwart dan kiest voor e5, dan kan wit weer kiezen uit 3 zetten (d4, Nf3 en Nc3). De drie puntjes betekenen dat deze zetten ook weer verwijzen naar mogelijke tegenzetten. Een zoekboom kan zeer snel vertakken in honderdduizenden (vanaf vijf plies[1]) of miljoenen unieke varianten (vanaf zes plies[1]).



Figuur 4: Een voorbeeld van een zeer eenvoudige zoekboom

2.2 Move ordering algoritmen

De miljoenen of miljarden varianten die worden gegenereerd door het move list algoritme kan men onmogelijk allemaal analyseren om de beste zet te vinden. De oplossing om het gigantisch aantal nodes[2] in een spelpositie terug te dringen naar een doorzoekbaar geheel, vond men in het alpha-beta pruning algoritme (zie 2.4 alpha-bate pruning algoritme) dat het aantal te doorzoeken varianten sterk kan terugdringen zonder het risico de beste variant over te slaan. De effectiviteit waarmee het alpha-beta pruning algoritme werkt, hangt echter sterk af van de volgorde van de varianten die worden onderzocht.

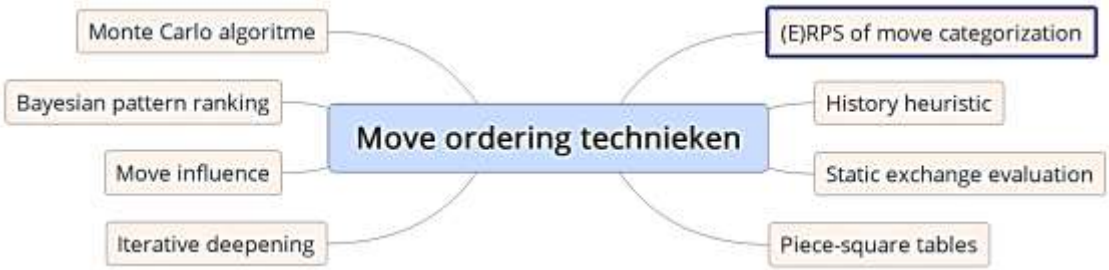
Het alpha-beta pruning algoritme kan een maximaal aantal nodes snoeien, als de beste zetten eerst worden behandeld (zie 2.4 Alpha-beta pruning algoritme). Zonder alle varianten te evalueren, weet men natuurlijk niet wat de beste beschikbare varianten zijn. Daarom heeft men een algoritme ontwikkeld dat alle beschikbare zetten analyseert en op basis daarvan een prognose maakt en de zetten sorteert van beste zet naar slechtste zet. De analyse die dit algoritme uitvoert moet aan twee voorwaarden voldoen: het moet zo accuraat en zo snel mogelijk zijn. Dit algoritme noemt men het move ordering algoritme. Men heeft verschillende soorten move ordering algoritmen afhankelijk van het soort techniek dat wordt gebruikt om de beschikbare varianten te analyseren.

[1] Ply: één zet van één speler
[2] Node: knoop in een zoekboom

Hoe beter de sortering gebeurt door het move ordering algoritme, hoe meer varianten het alpha-beta algoritme kan snoeien. De overblijvende varianten worden dan aan de evaluatiefunctie doorgegeven waarbij een doorgedreven analyse wordt uitgevoerd om de beste zet te bepalen.

De move ordering functie gaat dus voordat alle eindposities (leaf nodes[1]) in de zoekboom worden geëvalueerd zo goed mogelijk de varianten die door het move list algoritme werden gegenereerd sorteren van “best” naar “slechtst”.

Er zijn verschillende move ordering technieken waarvan we hier enkele kort bespreken. (Figuur 5: Enkele verschillende move ordering technieken). Veel schaakprogramma’s gebruiken een combinatie van verschillende move ordering technieken.



Figuur 5: Enkele verschillende move ordering technieken

2.2.1 History heuristic

History heuristic is een methode die werd ontworpen door Jonathan Schaeffer (8). Het gaat uit van het principe dat veel schaakposities in een game weinig van elkaar verschillen (bijvoorbeeld enkel de locatie van een bepaald stuk). Een goede zet in de ene positie zal hoogstwaarschijnlijk ook een goede zet zijn in de andere positie. Men telt dus hoeveel keren een bepaalde zet een cutoff[2] in een zoekboom veroorzaakt zonder rekening te houden met de positie waarin de zet gebeurt. De top zetten (killer moves) zijn de zetten die de meeste cutoffs[2] veroorzaken en worden bijgevolg eerst gekozen. Deze methode is zeer populair en wordt in alle vooraanstaande programma’s gebruikt. De techniek is geoptimaliseerd in 2006 door Mark Winands et al. (9) door de history heuristic relatief te maken door met gemiddelden te werken. Op die manier komen ook de bijna beste zetten aan bod. Deze verfijning leverde een winst op van tien tot vijftien procent in het denkspel LOA[3]. History heuristic is een bijkomende move ordering techniek als aanvulling of verbetering op een hoofdtechniek.

2.2.2 Static exchange evaluation

Voor elke variant wordt berekend wat de materiële score is (waarde stukken computer - waarde stukken tegenstander) na een reeks captures op hetzelfde veld. De zetten worden gesorteerd op basis van deze score. Om de score te bepalen, gebruikt men meestal de volgende waarden: één punt voor een pion, drie punten voor een loper of een paard, vijf punten voor een toren, negen punten voor een

[1] Een leaf node is een node zonder kinderen.(eindpositie van een variant of van een tak van een zoekboom)
[2] Cutoff: snoeien van een tak in een zoekboom
[3] Lines of Action: strategisch bordspel voor twee spelers.

koningin en een zeer hoge waarde (bijvoorbeeld 100) voor de koning. Deze vorm van move ordering wordt gebruikt als aanvulling op een hoofdtechniek. De vorm wordt ook zo diep doorgerekend als noodzakelijk (zolang als er recaptures mogelijk zijn) onafhankelijk van de zoekdiepte die is ingesteld om onnodig materiaalverlies te vermijden. Deze techniek wordt vooral gebruikt bij het bereiken van de gewenste diepte: men rekent tot op bepaalde diepte en men handelt dan alle lopende opeenvolgende captures af, zodat er geen valse winst of verlies wordt voorspeld (horizon effect).

2.2.3 Piece-square tables

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 5 & 10 & 15 & 20 & 20 & 15 & 10 & 5 \\ 4 & 8 & 12 & 16 & 16 & 12 & 8 & 4 \\ 3 & 6 & 9 & 12 & 12 & 9 & 6 & 3 \\ 2 & 4 & 6 & 8 & 8 & 6 & 4 & 2 \\ 1 & 2 & 3 & -10 & -10 & 3 & 2 & 1 \\ 0 & 0 & 0 & -40 & -40 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Figuur 6: Een mogelijke piece-square table voor de witte pionnen

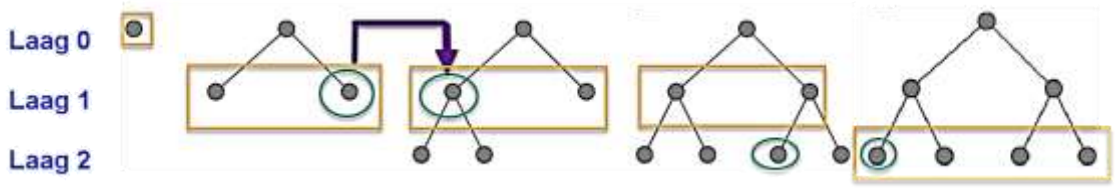
Bij deze methode worden er waarden toegekend aan de stukken en aan de locatie waar ze zich bevinden. De array in figuur 6 stelt de 64 schaakvelden van een bord voor. Voor elk veld bevat de array de waarde van het speelstuk als dat zich op dat veld bevindt. Zo wordt er een array aangemaakt voor elk stuk van elke kleur. Soms worden er verschillende tabellen aangemaakt voor de opening, het middenspel en het eindspel. Voor elke leaf node worden de waarden van alle stukken voor elke kleur op elk veld opgeteld. De score is uiteindelijk: score speler aan zet – score tegenpartij. De variant met de hoogste waarde is de beste variant. Deze techniek kan ook als evaluatietechniek worden gebruikt: Steve Maughan gebruikt piece-square tables als evaluatietechniek in zijn schaakprogramma Maverick (ELO[1]-Rating: 2.515) (11).

2.2.4 Iterative deepening

Iterative deepening is vooral populair in competitie schaken omdat deze techniek bijzonder geschikt is voor spelen met een tijdshandicap. Men plaatst de beste zet van een bepaalde laag als eerste te behandelen zet op de volgende laag zodat als er geen tijd meer over is om de volledige laag te onderzoeken men steeds de beste zet van de vorige laag reeds onderzocht heeft. Men heeft in het slechtste geval nog steeds de beste zet van de vorige laag ter beschikking als de berekening wordt onderbroken. Een voorbeeld aan de hand van figuur 7: men start bij laag 0. Dit is de eerste zet van wit en alle mogelijke zetten van wit worden gegenereerd in laag 1. In dit voorbeeld zijn er maar twee zetten mogelijk maar in werkelijkheid zijn er gemiddeld (in het middenspel) tweeëndertig zetten. De

[1] ELO-punten: geeft de sterkte van een speler weer. De sterkste menselijke speler heeft 2853 punten en het sterkste schaakprogramma 3393 punten

bekomen zetten op laag 1 worden geëvalueerd en de tweede zet blijkt de sterkste te zijn (groene cirkel). Deze tweede zet komt vooraan te staan en wordt volledig geëvalueerd tot op laag 2.

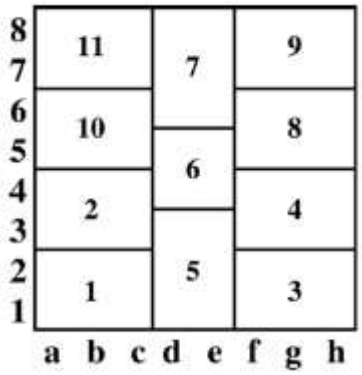


Figuur 7: Iterative deepening: het zoeken laag per laag

Vervolgens komt de volgende zet van laag 1 aan de beurt en deze wordt ook geëvalueerd tot op laag 2. Nu blijkt het eerste kind van deze zet de beste zet te zijn (groene cirkel). Wederom wordt deze zet vooraan geplaatst en wordt deze geëvalueerd tot op laag 3. Sommige programma's laten vanaf een bepaalde laag de minder goede zetten weg (late move reduction (LMR)). Men heeft dus te allen tijde een zet voorhanden als de tijd verstreken is (de zet met de groene cirkel). Men weet nooit hoe diep men gaat rekenen, men rekent gewoon tot de beschikbare tijd is verstreken. Een nadeel is de extra kost om alle lagen te evalueren. De techniek wordt veelal als evaluatietechniek gebruikt maar kan uiteraard ook als move ordering algoritme dienen.

2.2.5 Move influence

Deze techniek leert welke velden op het bord belangrijk zijn. Als een zet deze velden kan bestrijken, dan krijgt de zet voorrang. Een bord wordt hiervoor ingedeeld in zones (zie figuur 8). Deze techniek doet dienst als secundaire move ordering techniek in geval twee zetten eenzelfde beoordeling hebben gekregen in de primaire move ordering techniek. Het algoritme is verder uitgebreid door Kieran Greer (12) die ook rekening houdt met tactische zetten. Uit het onderzoek bleek dat het move influence algoritme meer nodes doorzocht dan de history heuristic techniek, maar veel meer tijd nodig heeft, zodat het in de praktijk minder bruikbaar is.



Figuur 8: Een voorbeeld hoe het bord kan worden ingedeeld in zones

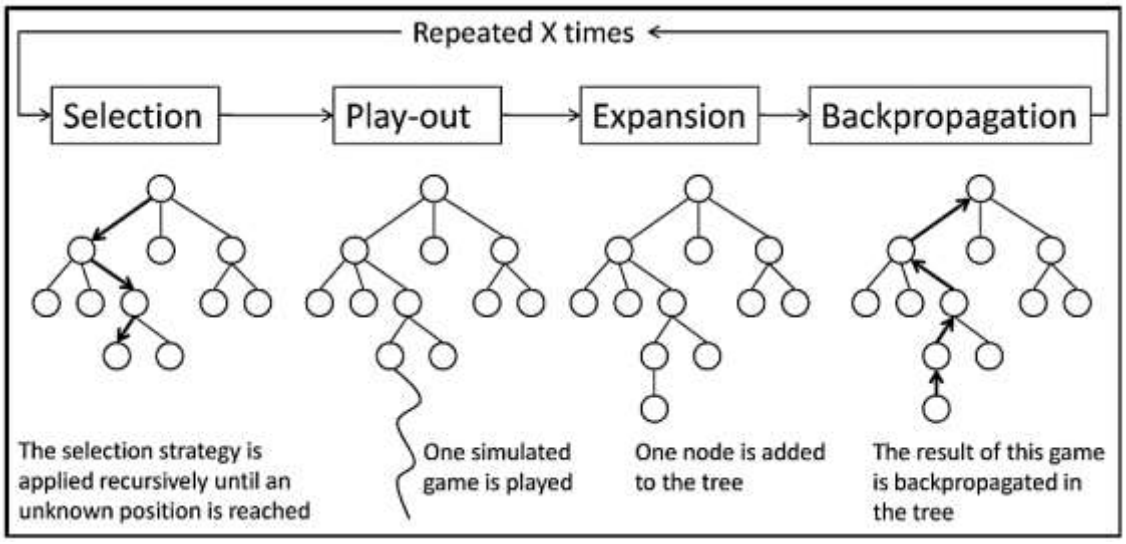
2.2.6 Bayesian pattern ranking

Deze methode gaat de zettenvolgorde proberen te voorspellen aan de hand van gespeelde partijen en een Bayesiaans leeralgoritme (13). Eerst worden een aantal generieke patronen onttrokken aan de database met gespeelde partijen. Deze patronen worden gedefinieerd als hash keys (unieke waarde die aan het patroon wordt gekoppeld). Het Bayesiaanse leeralgoritme gaat dan deze patronen vergelijken met spelposities. In het onderzoek van 2006 (13) heeft men een database gebruikt van

181.000 games (Go) van experten waaruit twaalf miljoen patronen zijn onttrokken. Het resultaat was dat het systeem in staat was in 34% van alle testposities de zet te voorspellen die professionele Go spelers zouden doen. Deze techniek heeft met Realization Probability Search gemeenschappelijk dat het gebruik maakt van eerder gespeelde toppartijen.

2.2.7 Monte Carlo algoritme

Het Monte Carlo algoritme (Figuur 9: Vierstappen plan van het Monte Carlo algoritme (8)) gaat vanuit een gegeven positie het spel duizenden keren met willekeurige zetten uitspelen. Op deze manier wordt een zoekboom in het geheugen opgebouwd en wordt de beste variant gekozen. Deze methode is al met succes toegepast in verschillende denkspelen (vooral in Go), maar nog niet in het schaken. Google scoorde in 2016 met AlphaGo (6) een gigantische succes (versloeg een Go-speler in de top vijf van de wereld) daarbij gebruik makend van het Monte Carlo algoritme. Dit algoritme kan ook als evaluatie techniek worden gebruikt. Deze techniek vereist wel zware hardware.



Figuur 9: Vierstappen plan van het Monte Carlo algoritme (8)

2.2.8 Transposition tables

Dit algoritme gaat alle informatie over elke positie die wordt onderzocht opslaan in een tabel als volgt: hashcode[1] van de positie, hoe diep dat de positie werd doorzocht en de score van de positie. Voordat een nieuwe positie wordt onderzocht, wordt er van die positie een hashcode gegenereerd. Als die hashcode in de tabel voorkomt, dan is deze positie reeds geëvalueerd tot op de diepte die staat aangegeven. Het resultaat is dat men veel minder varianten moet analyseren wat de move ordering een pak sneller doet werken. Deze techniek wordt ook gebruikt in de evaluatiefunctie.

2.2.9 Realization Probability Search

Deze move ordering techniek behandelen we in een apart hoofdstuk omdat deze techniek het onderwerp van dit onderzoek is.

[1] Hashcode: unieke code die wordt gegenereerd aan de hand van een object en dat object identificeert.

2.3 Evaluatiefunctie

De evaluatiefunctie bepaalt de waarde van een bepaalde positie altijd bij de leaf nodes[1]. Een leaf node in de zoekboom van figuur 4 is bijvoorbeeld d4 (linkse tak). De positie die de evaluatiefunctie in dat geval gaat evalueren is de positie die werd bereikt na e4-e5-d4. De evaluatie kan op verschillende manieren gebeuren van zeer eenvoudig tot zeer geavanceerd. De meest eenvoudige is het materiaalverschil te berekenen (waarde stukken wit – waarde stukken zwart). Geavanceerde evaluatiefuncties gaan ook rekening houden met positionele, tactische en strategische[2] kenmerken in een schaakstelling.

Zoals al eerder besproken, loopt het aantal te evalueren varianten snel op als men dieper wil rekenen en wordt het vanaf zes plies[3] diepte onmogelijk om alle varianten te berekenen binnen de toegestane tijd. Gelukkig kan men een groot aantal varianten schrappen door het gebruik van het alpha-beta pruning algoritme (zie 2.4 Alpha-beta pruning algoritme). Ook gaat men dubbele varianten, die zijn ontstaan doordat dezelfde zetten in een andere volgorde werden uitgevoerd en leidden tot eenzelfde positie, verwijderen (zie 2.2.8 transposition tables) . Op die manier kan men al snel enkele plies[3] dieper rekenen.

2.4 Alpha-beta pruning algoritme

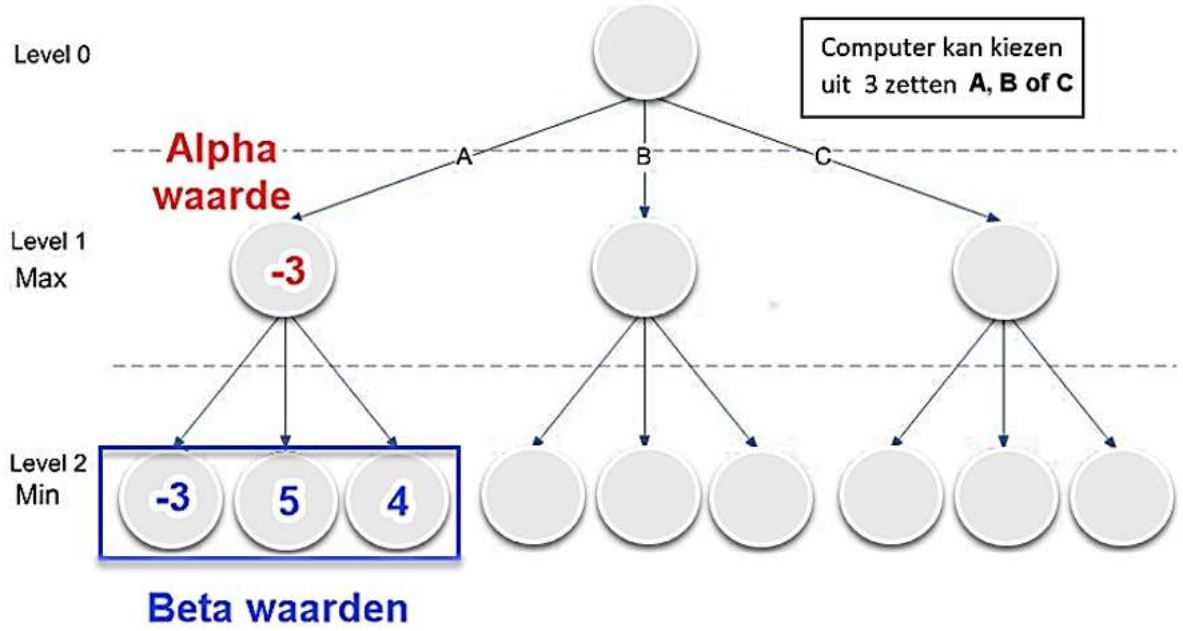
Het alpha-beta pruning algoritme zorgt ervoor dat we niet alle leaf nodes[1] moeten evalueren. Dit algoritme gaat sommige takken snoeien in de zettenboom zodat de leaf nodes van deze takken niet meer hoeven te worden geëvalueerd. Het aantal takken dat wordt gesnoeid, hangt af van hoe goed dat de move ordering zijn taak uitvoert. Het alpha-beta pruning algoritme kan namelijk een maximaal aantal nodes snoeien, als de beste zetten eerst worden geëvalueerd. De move ordering functie zorgt voor een prognose die de beschikbare zetten in een positie sorteert van best naar slechtst op basis van een prognose en deze zetten dan aanbiedt aan het duo alpha-beta pruning algoritme -evaluatiefunctie. Alle schaakprogramma's gebruiken de alpha-beta pruning techniek of een afgeleide daarvan. We leggen de werking van het alpha-beta pruning algoritme uit aan de hand van een voorbeeld:

We starten met een schaakpositie waar 3 zetten mogelijk zijn voor de computer (level 0): A, B en C (zie figuur 10: De eerste alpha waarde bepalen). Deze zetten zijn niet aangeboden geweest aan de move ordering functie en bevinden zich dus in een willekeurige volgorde. In level 0 is de computer aan zet en in level 1 de tegenpartij. De evaluatiefunctie start met de leaf nodes[1] van links naar rechts te evalueren. We berekenen de waarden van de eerste groep kinderen die we de betawaarden zullen noemen. Alle waarden drukken altijd de waarde van de positie uit ten overstaan van de witspeler (in dit geval de computer). Hoe negatiever, hoe slechter voor de witspeler.

De minst gunstige waarde (-3) wordt aan de ouder toegekend omdat we ervan uitgaan dat de tegenpartij (die aan zet is in level 1) natuurlijk de minst gunstige zet voor de computer zal kiezen (dat

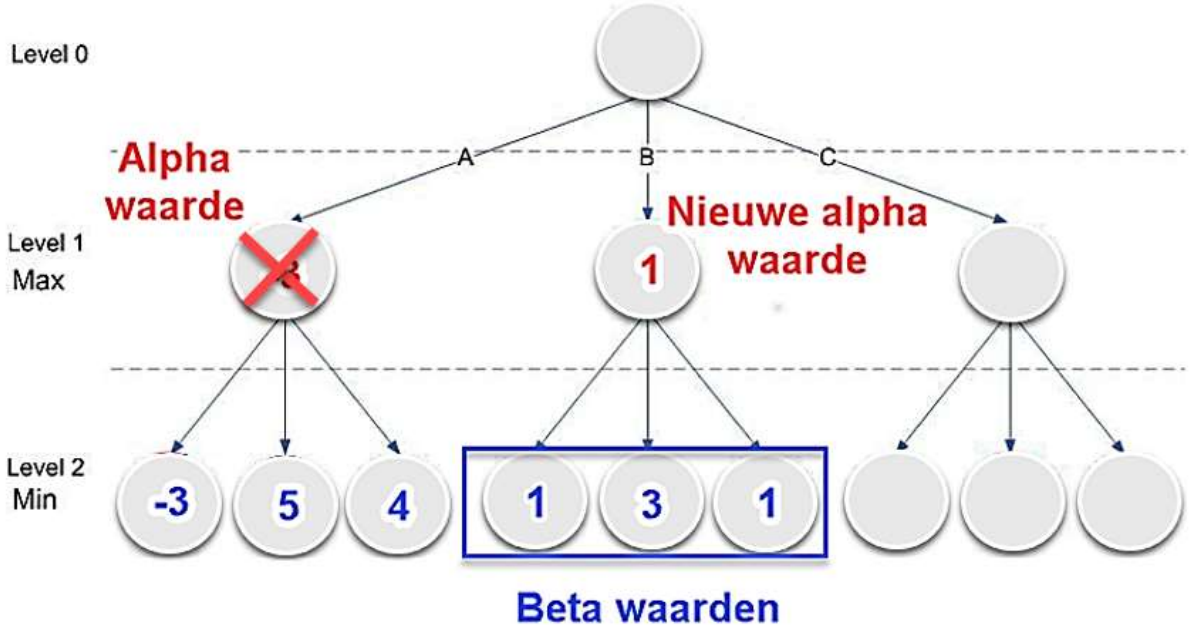
[1] Een leaf node is een node zonder kinderen. (eindpositie van een variant)
[2] Zie bijlage 2: verklarende woordenlijst
[3] Ply: één zet van één speler

is dan uiteraard de beste zet voor de tegenpartij). We noemen deze waarde de alphawaarde. De node met de hoogste alpha waarde zal uiteindelijk door de computer worden gekozen als zet.



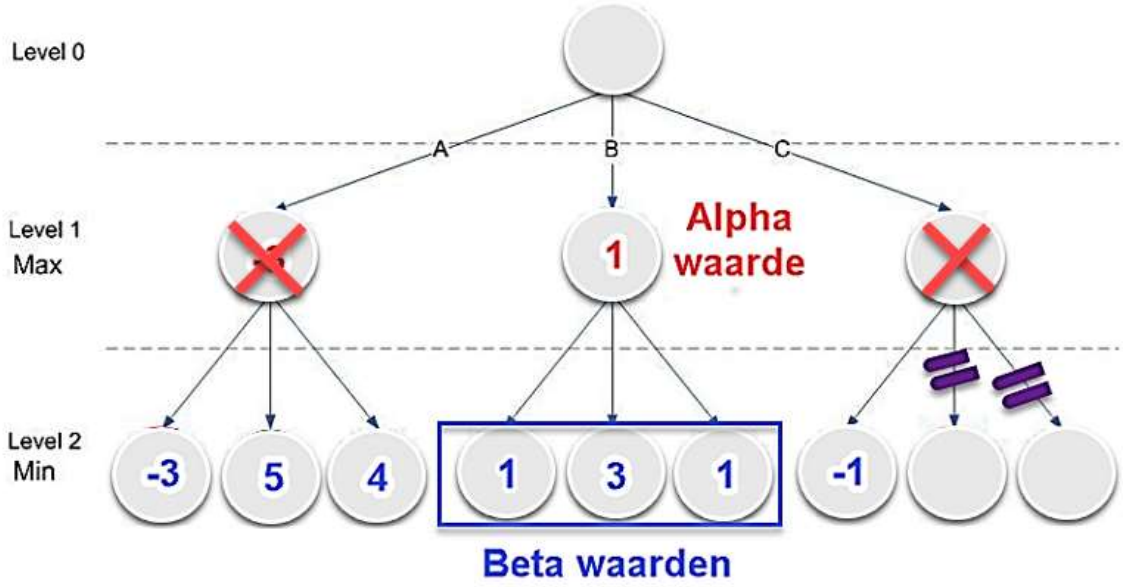
Figuur 10: De eerste alpha waarde bepalen

Vervolgens wordt het eerste kind van de tweede ouder geëvalueerd en we verkrijgen de waarde 1 (zie figuur 11). We vergelijken deze betawaarde met de vorige bekomen alphawaarde die gelijk is aan -3. De betawaarde is gunstiger voor het schaakprogramma dan de reeds bekomen alphawaarde en voorlopig zal het computerprogramma dus zet B spelen in plaats van zet A om de tegenstander geen kans te geven om een betere tegenzet te doen. Vervolgens worden de twee andere kinderen geëvalueerd en telkens is de bekomen waarde gunstiger dan de huidige alphawaarde dus blijft de alpha waarde op 1 staan. De tegenpartij kiest immers de minst gunstige zet voor de computer (zie figuur 11).



Figuur 11: De tweede groep levert een nieuwe alpha waarde op

De nieuwe alpha waarde is hoger dan de vorige alphawaarde en wordt in dat geval de nieuwe alpha waarde omdat de computer sowieso voor deze zet zal kiezen om de minder gunstige variant van zet A te vermijden. Dan komt de derde groep aan de beurt (Zie figuur 12: Beperking zoekboom door het



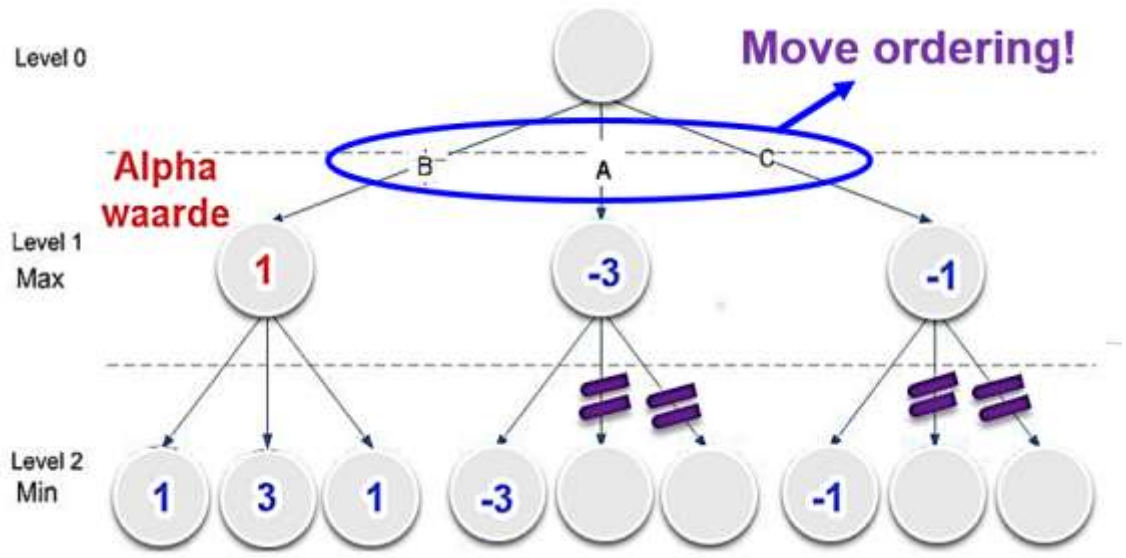
Figuur 12: Beperking zoekboom door het alpha beta pruning algoritme

alpha beta pruning algoritme). Het eerste kind van deze groep levert de betawaarde -1 op. We vergelijken deze weer met de nieuwe alphawaarde die gelijk is aan 1. De alphawaarde is nu groter dan de betawaarde. Dit betekent nu dat we de andere kinderen van groep 3 niet meer moeten evalueren, want het schaakprogramma gaat nooit zet C in aanmerking nemen. Er is immers één variant die altijd kan worden gespeeld door de tegenstander waarbij het schaakprogramma minder goed uit de verf komt dan bij alle varianten van groep 2. Dus heeft het geen zin om de resterende twee posities te evalueren, hoe fantastisch ze ook voor de computer mogen zijn. We zien dat we hier het aantal oorspronkelijke evaluaties terug gedrongen hebben van negen naar zeven. Dit betekent een winst van tweeëntwintig percent. We kunnen dus de resterende takken van een branch snoeien als beta < alpha.

Stel dat we terug opnieuw beginnen maar nu eerst het move ordering algoritme zijn werk laten doen. De prognose van het move ordering algoritme leidt tot de volgende volgorde van mogelijke zetten: B – A – C. We herhalen nu dezelfde procedure. Nu blijkt dat we vier takken kunnen snoeien (Figuur 13: Als de move ordering er in slaagt om de beste zet eerst te zetten, kan er maximaal worden gesnoeid!).

We hebben nu het aantal evaluaties teruggedrongen van negen naar vijf. Dit betekent een winst van vierenveertig percent! In het slechtste geval (de beste zet staat laatst) levert alpha-beta pruning geen winst op. In het beste geval (de beste zet staat eerst) hoeft men (buiten de eerste tak) enkel het eerste kind van elke tak te evalueren. Men kan dus alpha-beta pruning toepassen zonder het gevaar de beste variant over het hoofd te zien.

Het alpha-beta pruning algoritme heeft dankzij het move ordering algoritme de zoekboom met 44% kunnen terugdringen!



Figuur 13: Als de move ordering er in slaagt om de beste zet eerst te zetten, kan er maximaal worden gesnoeid!

Het theorema van Levin (14) bepaalt hoeveel leaf nodes[1] we werkelijk moeten evalueren bij het toepassen van alpha beta pruning:

Bij een oneven aantal plies[2] (diepte):

$$T = b^{\frac{n+1}{2}} + b^{\frac{n-1}{2}} - 1$$

Bij een even aantal plies[2]:

$$T = 2b^{n/2} - 1$$

Waarbij T = aantal te evalueren leaf nodes, b = aantal branches per leaf node en n = aantal plies[2] (diepte). Toegepast op ons voorbeeld:

$$T = 2 * (3)^{2/2} - 1 = 5$$

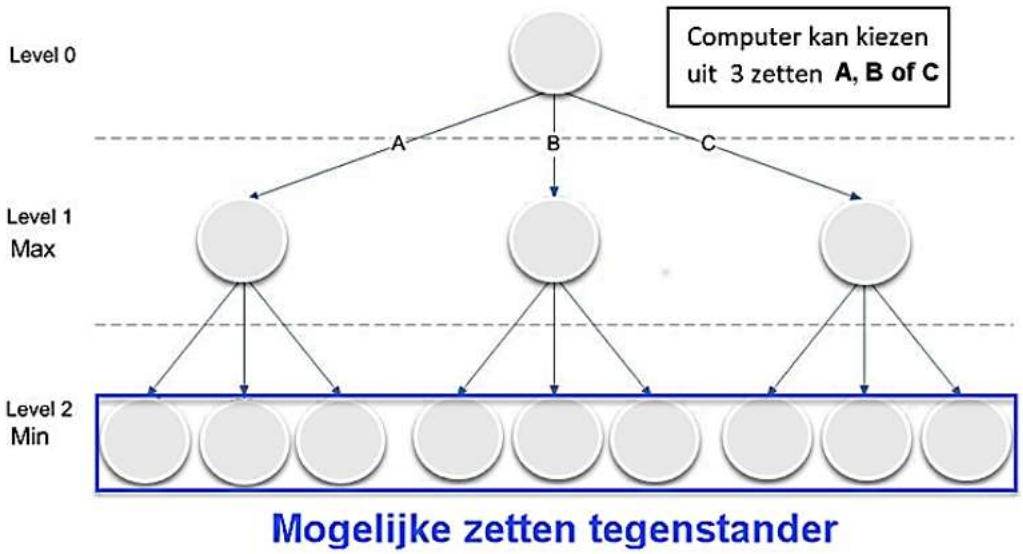
Het is dus heel belangrijk dat de beste zet zo snel mogelijk wordt geëvalueerd. De move ordering functie moet de beschikbare zetten zo goed mogelijk sorteren van best naar slechtst op basis van een prognose en deze voorspelling dan doorgeven aan het alpha-beta pruning algoritme.

2.5 Search algoritme

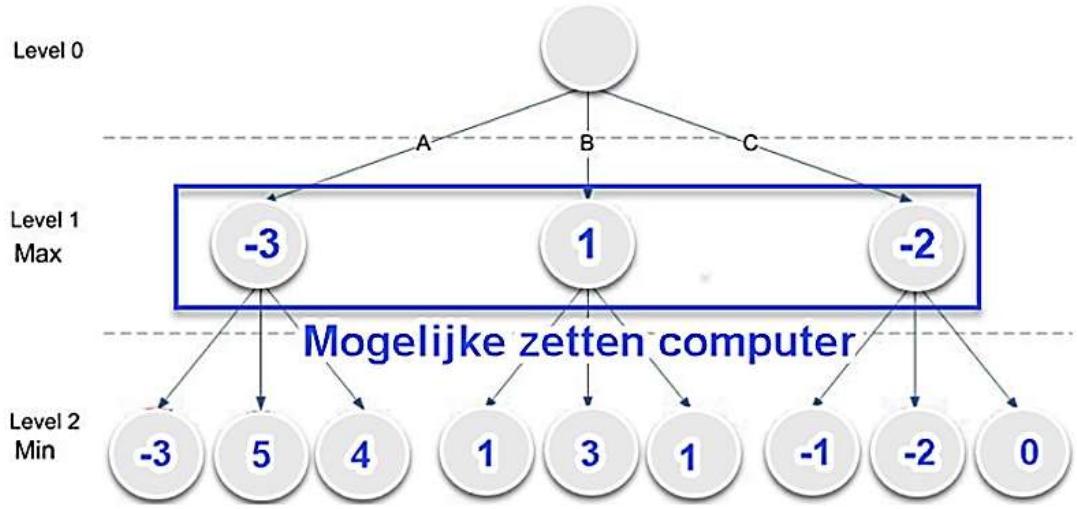
Het search algoritme start zijn taak als de evaluatiefunctie klaar is met de leaf nodes te evalueren. Het search algoritme zoekt de meest gunstige variant, namelijk de hoofdvariant (PV – principal variation). De eerste zet van die variant is dan de zet die de computer uiteraard zal spelen. In theorie is het best de PV te laten leiden naar de eindpositie in het spel, maar praktisch gezien is dit momenteel nog

[1] Leaf node: node zonder kinderen
[2] Ply: één zet van één speler

onmogelijk wegens de complexiteit in het schaken (1.4 Complexiteit van het schaakspel). In bijna alle schaakprogramma's gebeurt het bepalen van de PV door het minimax algoritme. Een praktisch voorbeeld (zie figuur 14): het evaluatie algoritme start met de leaf nodes[1] te evalueren van links naar rechts. In het voorbeeld wordt er twee plies[1] diep gerekend. Maar ook als het programma tien plies[1] diep rekent, start het algoritme altijd met de leaf nodes. Nadat het evaluatiealgoritme zijn taak heeft gedaan, ziet het resultaat eruit als in figuur 15: De resultaten van de evaluatiefunctie.



Figuur 14: Het minimax algoritme toepassen

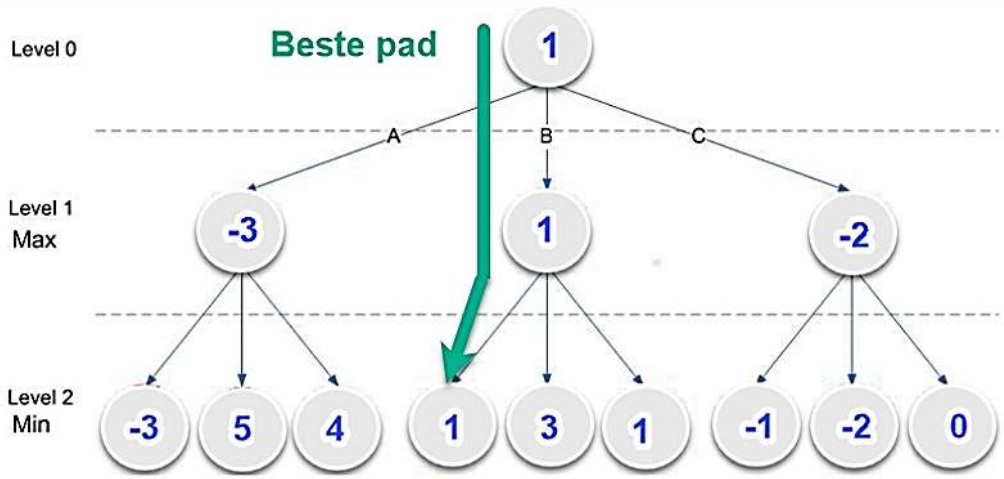


Figuur 15: De resultaten van de evaluatiefunctie

Nadat alle nodes in level 1 een waarde hebben gekregen, schiet het minimax algoritme in actie. De ouder in level 0 krijgt de maximumwaarde toegekend van zijn kinderen, omdat in level 1 de computer aan zet is en dus voor de sterkste zet (MAX) wordt gekozen. Het beste pad is nu bekend door te

[1] Ply: één zet van één speler

vertrekken van de top node naar beneden afwisselend kiezend voor de maximum- en minimumwaarde[1] (zie 2.5 Search algoritme): level 1 (max) = 1 en level 2 (min) = 1.



Figuur 16: Het bepalen van de hoofdvariant

2.6 Game list algoritme

Dit is eigenlijk een zeer eenvoudig algoritme dat een lijst bijhoudt van de gespeelde zetten en de variant die op dat moment wordt onderzocht.

[1] Als aan twee nodes in een branch dezelfde waarde is toegekend, kan men een bijkomende evaluatie doen (bv. positionele stellingskenmerken) om een keuze te maken of men kan ook simpelweg een random keuze maken.

3. Onderzoeksmodel

In deze thesis onderzoeken we hoe we de huidige schaakprogramma's nog sterker kunnen laten spelen door een move ordering algoritme te ontwikkelen dat gebruik maakt van de expertise die zit opgeslagen in een database van meer dan 3.300 partijen op topniveau. Een move ordering algoritme sorteert zo goed en zo snel mogelijk de beschikbare zetten in een positie van best naar slechtst, waarna de prognose wordt doorgegeven aan het duo alpha-beta pruning algoritme – evaluatiefunctie. Hoe beter de prognose, hoe meer varianten het alpha-beta pruning algoritme kan schrappen (zie 2.4 Alpha-beta pruning algoritme) tijdens het evalueren van de beschikbare zetten.

We hanteren een techniek die gebruik maakt van statistische data namelijk: realization probability search. Eerder werd deze techniek gebruikt in een Shogi programma waarbij de move ordering met een spectaculaire 72% verbeterde (3). Kan RPS[1] ook succesvol in een schaakprogramma dienst doen als move ordering algoritme?

De onderzoeksvraag luidt als volgt:

Wat is de bruikbaarheid van RPS[1] in een schaakprogramma als move ordering techniek?

De bruikbaarheid van een move ordering techniek wordt bepaald door twee kenmerken:

1. Voorspellingsgraad: hoe correcter de voorspellingsgraad, hoe meer varianten dat het alpha-beta pruning algoritme kan snoeien en hoe meer tijd vrijkomt om de bepaalde varianten dieper uit te rekenen.
2. Snelheid: hoe sneller er een prognose wordt bereikt, hoe meer tijd vrijkomt om (bepaalde) zetten dieper uit te rekenen.

Om de onderzoeksvraag correct te kunnen beantwoorden, moeten we dus twee deelvragen beantwoorden:

- Wat is de voorspellingsgraad van RPS[1] vergeleken met andere move ordering technieken in een schaakprogramma?
- Wat is de snelheid van RPS[1] vergeleken met andere move ordering technieken in een schaak-programma?

Om de bruikbaarheid van RPS[1] als move ordering techniek te testen in een schaakprogramma volgen we het volgende plan:

1. Opstellen van de verschillende zetcategorieën.
2. Bepalen van de probability index van deze zetcategorieën.
3. Implementeren van het move ordering algoritme op basis van een ontworpen probability tabel.
4. Het move ordering algoritme testen en de bekomen resultaten vergelijken met de resultaten van een klassiek move ordering algoritme. Daarvoor dienen we eerst een klassiek move ordering algoritme te implementeren.
5. De bekomen resultaten vergelijken met resultaten die teruggevonden worden in de literatuur.

[1] Realization probability search

Bij het opstellen van de lijst met zetcategorieën worden enkel die categorieën opgenomen die informatie bevatten over strategische (16)(17), tactische (18)(19), positionele (20)(21) of materiële eigenschappen die voor of na de zet aanwezig zijn in de schaakstelling. Deze eigenschappen wijzen allemaal op een voor- of nadeel in een schaakpositie.

Om de lijst zo compleet mogelijk te maken, voeren we een onderzoek uit in de schaakliteratuur naar wat de belangrijkste strategische, tactische, positionele en materiële eigenschappen zijn die een zet of een schaakpositie kan bevatten.

Om de kanspercentages van deze zetcategorieën te bepalen, ontwerpen we software die meer dan 3.300 top schaakpartijen analyseert en daarbij vaststelt hoeveel keer elke categorie wordt gespeeld en hoeveel keer die categorie kon worden gespeeld in die partijen.

De zetcategorieën samen met hun kanspercentages vormen dan de probability tabel waarbij achter elke categorie een percentage staat: de kans dat die categorie zal worden gespeeld indien deze categorie kan worden gespeeld in een bepaalde schaakpositie.

Deze probability tabel vormt dan de hoeksteen om RPS[1] te implementeren. De software moet voor elke zet van elke partij een gesorteerde lijst van de beschikbare zetten kunnen genereren door middel van RPS[1]. We maken twee verschillende versies: een versie waarbij enkel rekening wordt gehouden met de probability van de zet en een versie met een opdeling in vier zetcategorieën die we voorrang geven op elkaar: captures, dringende , voordelige en de rest van de zetcategorieën. Bij de tweede variant vermenigvuldigen we de probability van een zet met een voorrangsfactor die is toegekend aan de categorie waartoe de zet behoort.

Om de bruikbaarheid van RPS[1] te kunnen beoordelen, vergelijken we ons move ordering algoritmen op basis van RPS[1] met een klassieke move ordering variant namelijk: piece square tables. We hebben voor deze techniek gekozen omdat piece-square tables gebruik maakt van twee evaluatiemethoden die in de schaakwereld als zeer belangrijk worden aanschouwd namelijk: de materiële score en de bezetting van het veld. Om zeker te zijn dat dit algoritme geen combinatie is van verschillende technieken, maken we eigen implementatie aan.

Als testmethode laten we alle implementaties van de move ordering systemen (RPS[1] en piece square tables) alle posities beoordelen die voorkomen in twintig testgames die werden gespeeld tussen schaakprogramma's die minstens een ELO[2]-rating hebben van 3.000 punten. Ter vergelijking: de huidige menselijke wereldkampioen Magnus Carlsen heeft een ELO[2]-rating van 2.838 punten. De prognoses van beide systemen vergelijken we dan met de zet die werkelijk in de partij werd gespeeld. Deze zet is voor ons de beste zet vermits de zet is uitgerekend door de sterkste schaakprogramma's ter wereld.

Om de resultaten van ons move ordering algoritme nog beter te kunnen beoordelen en wegens gebrek aan een gouden standaard vragen we aan bekende auteurs van schaakprogramma's om hun werkwijze en resultaten van hun move ordering systeem mee te delen.

Vervolgens trekken we een conclusie over de bruikbaarheid van RPS[1] in een schaakprogramma.

[1] Realization probability search

[2] ELO-punten: geeft de sterkte van een speler weer. De sterkste menselijke speler heeft 2853 punten en het sterkste schaakprogramma 3393 punten

4. Probability tabel

4.1 Zetcategorieën

De probability tabel vormt het hart van ons move ordering algoritme dat gebruik maakt van realization probability search. Deze probability tabel bevat zetcategorieën die informatie bevatten over strategische, tactische, positionele of materiële eigenschappen die voor of na de zet aanwezig zijn in de schaakstelling. Deze eigenschappen wijzen allemaal op een voor- of nadeel in een schaakpositie. Voor elke zetcategorie is ook de probability opgenomen dat deze categorie in een willekeurige schaakpositie zal worden gespeeld, als deze categorie mogelijk is in die positie.

Strategische eigenschappen van een schaakpositie gaan vooral over de veldbeheersing. Men kan bijvoorbeeld de eigen paarden ruilen met de lopers van de tegenpartij. Men heeft dan een strategisch voordeel omdat op termijn de lopers meer velden kunnen bestrijken dan de paarden. Hoe meer stukken er afgeruild worden, hoe krachtiger het loperpaar wordt. Andersom, als men de lopers afruilt tegen de paarden heeft men een strategisch nadeel.

Tactische eigenschappen zijn voor- of nadelen op korte termijn. Men heeft bijvoorbeeld een tactisch voordeel als men via een geforceerde combinatie materiaal kan winnen of de tegenstander kan mat zetten. Tactische zetten zijn bijvoorbeeld aftrekschaak, dubbele aanval, penning, … .

Positionele eigenschappen wijzen op de structuur van de schaakpositie, bijvoorbeeld: een aaneenschakeling van pionnen (de pionnen verdedigen elkaar), een vrijpion (kan niet worden geslagen door een vijandelijk pion), een geïsoleerde pion (kan niet meer worden verdedigd door een eigen pion), de koning die sterk staat verdedigd, een open lijn waarlangs de zware stukken (dame en torens) kunnen manoeuvreren.

De materiële eigenschappen van een schaakpositie geven informatie over het materiaal dat aanwezig is op het bord. Een voorbeeld is de verhouding tussen de waarde van de witspeler zijn stukken en de waarde van de zwartspeler zijn stukken. Bij de waardebepaling van een stuk wordt voor volgende verdeling gekozen: 9 punten voor een dame, 5 punten voor een toren, 3 punten voor een loper of paard en 1 punt voor een pion. Een paard afruilen tegen een toren, levert dus materieel voordeel op.

We hebben om de lijst zo compleet mogelijk te maken in de schaakliteratuur strategische (16)(17), tactische (18)(19), positionele (20)(21) en materiële (22) kenmerken opgezocht die een stelling kan bevatten en daarmee een lijst van categorieën opgesteld[1]. De uiteindelijke lijst bevat 124 categorieën. De volledige tabel met zetcategorieën vindt u in bijlage 3.

4.2 Probability index

De probability index van een zetcategorie is de kans dat een zetcategorie *zal* worden gespeeld indien deze categorie *kan* worden gespeeld in een bepaalde schaakpositie. Om de probability index te bepalen van de opgestelde zetcategorieën hebben we schaakpartijen gebruikt die openbaar te vinden zijn op het internet (14). We gebruikten een partijenbestand met 3.350 partijen die zijn gespeeld door

[1] Zie bijlage 3: zet categorieën

Komodo (wereldkampioen schaakprogramma's – 3.394 ELO[1] punten) tegen de rest van de wereld (computerprogramma's met minstens 3.000 ELO[1] punten).

Om de probability index van deze zetcategorieën te bepalen, hebben we software nodig die de schaakregels kent, routines aan boord heeft om in een schaakpositie alle beschikbare legale zetten te genereren en kan omgaan met het PGN formaat. Het PGN formaat staat voor Portable Game Notation (PGN). Het is een vorm van schaaknotatie waarbij de gegevens van schaakpartijen worden opgeslagen in platte tekstbestanden. PGN vormt de standaard voor het vastleggen van partijen. Praktisch elk schaakprogramma en elke schaakdatabase ondersteunt het PGN formaat. We maken gebruik van de open source bibliotheek Chesspresso[2] omdat deze bibliotheek voldoet aan al deze voorwaarden. We zorgen ervoor dat ons programma in eerste instantie een PGN bestand inleest en alle partijen die zich in dat bestand bevinden filtert:

- Alle dubbele partijen worden verwijderd.
- Alle snelschaakpartijen worden verwijderd.
- Alle blindschaakpartijen worden verwijderd.
- Alle internet partijen worden verwijderd.
- Alle partijen met minder dan vijftien zetten worden verwijderd.
- Alle zetten worden gecontroleerd op geldigheid. Corrupte partijen worden verwijderd.

De reden hiervoor is dat we enkel unieke kwaliteitspartijen willen overhouden, wat bijvoorbeeld niet het geval is met snelschaakpartijen. Het programma analyseert dan alle posities, die voorkomen in een partij, en gaat na welke categorieën er op dat moment konden worden gespeeld en welke categorie er werkelijk is gespeeld. Na het verzamelen van deze informatie wordt voor elke categorie de probability index berekend en wordt de probability index tabel gegenereerd (zie figuur 17: voorbeeld probability index tabel). De volledige tabel vindt u in bijlage 5.

ID	Categorieën	Probability Index %
1	Mat	100,00
2	Stuk (vertrek) geen aanvalszet	89,21
3	Geen Capture	83,91
4	Stuk (vertrek) geen verdedigingszet	81,43
5	Stuk (aankomst) geen verdedigingszet	78,44
6	Stuk (aankomst) kan niet worden geslagen	76,87
7	Stuk (vertrek) kan niet worden geslagen	76,44
8	Zet vermeerdert mobility	76,28
9	Stuk (vertrek) kan worden geslagen door minder stuk	70,64
10	Zet vermindert mobility	63,85
11	Stuk behoudt zicht op het centrum	62,27

Figuur 17: Gedeelte van de gegenereerde probability tabel

[1] ELO-punten: geeft de sterkte van een speler weer. De sterkste menselijke speler heeft 2853 punten en het sterkste schaakprogramma 3393 punten

[2] http://www.chesspresso.org/

5. Realization probability search

5.1 Achtergrond

Realization probability search (RPS) is een move ordering techniek die succesvol is getest in andere denkspelen(2)(3), maar nog niet in een schaakprogramma. RPS[1] bepaalt de move ordering door gebruik te maken van een probability tabel die de speelkansen bevat van mogelijke zetcategorieën. De kans dat een variant wordt gespeeld in een bepaalde positie, wordt bepaald door het product van de probabilities van de zetcategorieën waartoe de zetten van die variant behoren.

De realization probability index van een positie geeft aan hoe groot dat de kans is dat de variant die tot deze positie leidt, werkelijk wordt gespeeld. De realization probability van een positie wordt als volgt berekend (3):

Figuur 18: Het berekenen van de probability van een bepaalde positie

Waarbij

P_x = realization probability van een positie x (nieuwe positie)
P_c = transition probability voor een zet m die positie x verandert naar positie x' (category probability)
$P_{x'}$ = realization probability van de vorige positie (startpositie)

De kans bijvoorbeeld dat een pad wordt gekozen waar een koningin wordt geruild tegen enkel een pion is bijna onbestaande. Deze zet heeft een realization probability index van nagenoeg nul. Hier moet dan ook geen kostbare tijd aan worden verspeeld.

Het pad met de grootste realization probability bevat de beste zet en kan eventueel dieper worden doorgerekend (enhanced RPS[1]).

De transition probability index van een zet wordt bepaald door de categorie waartoe de zet behoort (P_c). Deze categorieën zijn bijvoorbeeld: schaakgevende zet, slaan van een onverdedigd stuk, een pion die promoveert tot dame, …. De probability index van elke categorie wordt bepaald door professionele partijen te analyseren en door volgende formule toe te passen:

[1] Realization Probability Search

$$P_c = \frac{n_p}{n_c}$$

Waarbij

P_c = de kans dat een zet van deze categorie wordt gespeeld (transition probability index).

n_p = aantal posities waarin een zet van deze categorie is gespeeld (in de expert partijen).

n_c = aantal posities waarin zetten van categorie c legaal zijn (in de expert partijen).

Als een zet tot meerdere categorieën behoort, dan wordt het gemiddelde van de probabilities van die categorieën genomen.

De samengestelde kansverdeling van een variant die een opeenvolgend n aantal zetten bevat (diepte) met RPS[1], is als volgt:

RP(s) = 1
RP (zet 1) = RP(s) * P (zet 1)
RP (zet 1, zet 2) = RP (zet 1) * P (zet 2)
RP (zet 1, zet 2, zet 3) = RP (zet 1, zet 2) * P (zet 3)
RP (zet 1, zet 2, …, zet n) = RP (zet 1, zet 2, …, zet (n-1)) * P (zet n)

Waarbij s= startpositie
RP= realization probability index van de positie
P = probability index van de categorie waartoe de zet behoort.

Higashiuchi en Grimbergen (15) hebben het effect van RPS[1] onderzocht in het spel Amazons en concludeerden dat het programma sterker speelt met RPS[1]. Het effect van RPS[1] werd nog sterker door rekening te houden met de fase van het spel waarin men zich bevond.

In een onderzoek van Tsuruoka et al. (3) wordt RPS[1] uitgetest op Shogi waar het leidde tot een verdubbeling van de zoeksnelheid. Deze snelheid wordt bekomen omdat aan onrealistische zetten heel weinig rekenkracht wordt besteed.

Winands en Bjornsson (2) hebben een verbeterde (enhanced) variant van RPS[1] toegepast op het spel Lines of Action. De verbetering zorgt voor een variabele zoekdiepte door rekening te houden met de rangorde van de zet in de zoekboom. De eerste zetten worden tot op minimumdiepte gerekend (beste zetten) om vroege cutoffs[2] te vermijden. Dit houdt meer rekening met tactische zetten die dikwijls over het hoofd worden gezien en vermijdt dat er te veel tijd aan slechte zetten wordt gespendeerd. Figuur 18 geeft een overzicht van de resultaten van het onderzoek.

Pairings	Score	Winning ratio
RPS vs. Classic	375-225	1.67
ERPS vs. Classic	433-167	2.59
ERPS vs. RPS	372.5-227.5	1.63

Figuur 19: Testresultaten van (E)RPS

[1] Realization probability search
[2] Cutoff: snoeien van een tak in de zoekboom

5.2 Algoritme

De RPS[1] software die we hebben ontwikkeld, gaat aan de hand van de ingebouwde probability tabel de move ordering bepalen van alle mogelijke legale zetten in elke schaakpositie van elke partij die in het aangeboden PGN[2] bestand zit.

De RPS[1] software genereert alle mogelijke varianten van een bepaalde schaakpositie via een recursief algoritme waarbij op elk niveau rekening wordt gehouden met de drempelwaarde. Van elke variant wordt de RPS[1] waarde bepaald. De RPS[1] waarde van een variant is het product van de probability indexen van de zetten die deze variant bevat. De drempelwaarde geeft de laagste RPS[1] waarde aan waarbij een variant wordt behouden. Als de RPS[1] waarde van een variant onder de drempelwaarde valt, wordt met deze variant geen rekening meer gehouden.

Voor elke zet in een variant wordt de probability index berekend aan de hand van de gegenereerde probability tabel. De RPS[1] waarde van een zet is het gemiddelde van de probability indexen van de categorieën waartoe die zet behoort.

Varianten die niet voldoen aan de materiaalscore worden ook genegeerd. Om de materiaalscore te bepalen, gebruiken we de volgende waarden voor de schaakstukken: 9 voor een dame, 5 voor een toren, 3 voor een paard en een loper en 1 voor een pion.

De overblijvende varianten worden dan gesorteerd op basis van hun RPS[1] score. De variant met de hoogste score zal als beste zet worden aanzien. De variant die uiteindelijk is gespeeld in de partij krijgt ook een ranking toegewezen op basis van zijn RPS[1] score. De uitvoer bij een bepaalde positie die drie levels diep is gerekend, ziet er bijvoorbeeld als volgt uit:

Move ordering voor ply 36: Kxd7

Aantal gegenereerde reeksen: 30416
Aantal gefilterde reeksen (RPS waarde minder dan 0.47): 5742
Aantal gefilterde reeksen (MateriaalScore minder dan -2.0): 24529
Aantal overblijvende reeksen op 3 levels diep: 145

Ranking: 1: Nxh2 (0.72)-Nxf8 (0.98)-Rxf8 (1.1)-
RPS Score: 0.78
Materiaal Score: -2.0

Gespeelde zet in partij:
Ranking: 7: Kxd7 (0.75)-Nd4 (0.71)-Nxe5 (1.08)-
RPS Score: 0.58
Materiaal Score: 1.0

Het aantal gegenereerde, gefilterde en onderzochte reeksen wordt weergegeven. Aan de hand van de RPS[1] score (0.78) wordt de zet Nxh2 als beste zet aanzien. In de partij is echter Kxd7 gespeeld. Omdat de RPS[1] score van deze variant (0.58) lager ligt dan de RPS[1] score van Nxh2 ligt de ranking lager.

[1] Realization probability search
[2] Portable game notation

Tevens wordt ook de ranking aangegeven van de zet die uiteindelijk is gespeeld. Deze zet wordt aanschouwd als de beste zet die mogelijk is, omdat de partijen die we aanbieden aan onze implementatie, partijen zijn tussen twee schaakprogramma's met een ELO[1] rating van minstens 3.000. Aangezien de beste zet tot op heden nog onbekend is, omdat een schaakpartij momenteel nog niet volledig kan worden doorgerekend, kunnen we stellen dat de sterkste schaakprogramma's de beste zet leveren.

De ontworpen RPS[2] software maakt gebruik van enkele argumenten:

- **Zoekdiepte**: geeft aan hoeveel plies[3] diep er moet worden gezocht.
- **Startmove**: geeft aan bij welke ply[3] moet worden gestart met de move ordering.
- **Endmove**: geeft aan bij welke ply[3] moet worden gestopt met de move ordering. (Voor alle zetten geef je een 0 in)
- **Drempelwaarde**: geeft de cutoff[4] waarde (tussen 0 en 1) aan. (Alle zetten met een probability index lager dan de cutoff[4] waarde wordt geen rekening mee gehouden.) Dit kan als gevolg hebben dat met een goede zet die in eerste instantie niet goed werd beoordeeld, geen rekening wordt gehouden. Indien je geen cutoff [4] waarde wilt opgeven en rekening wilt houden met alle zetten, dien je 0 op te geven.
- **minMateriaalScore**: geeft het gewenste niveau van materiaalwinst aan op de gewenste zoekdiepte. 0 betekent dat er geen materiaalwinst is voor beide partijen.

[1] ELO-punten: geeft de sterkte van een speler weer. De sterkste menselijke speler heeft 2853 punten en het sterkste schaakprogramma 3393 punten

[2] Realization probability search

[3] Ply: één zet van één speler

[4] Cutoff: snoeien van een tak in de zoekboom

6. Klassieke move ordering (piece square tables)

Om de resultaten van RPS[1] te kunnen vergelijken met de resultaten van een ander move ordering algoritme, hebben we een mail gestuurd naar alle auteurs van de bekendste schaakprogramma's met de vraag of ze wilden meedelen welk move ordering algoritme ze gebruikten in hun programma samen de snelheid en de voorspellingspercentage. Alle auteurs, uitgezonderd Matthew Lai[2], weigerden hier aan mee te werken. De reden hiervoor is dat er een hevige concurrentiestrijd woedt tussen de verschillende auteurs met hun schaakprogramma. Zij nemen immers allemaal deel aan het wereldkampioenschap schaken voor computers en zijn bang dat hun algoritme de tegenstander hun schaakprogramma sterker zal maken.

Wegens gebrek aan vergelijkingsmateriaal en een gouden standaard hebben we een klassiek move ordering algoritme 'piece-square tables' geïmplementeerd om toch de resultaten van RPS[1] in een bepaald perspectief te plaatsen.

We hebben voor piece-square tables gekozen omdat deze techniek gebruik maakt van twee belangrijke evaluatiemethoden, namelijk: de materiële score en de bezetting van het veld.

De waarde van een stuk hangt af van het type stuk (koning, dame, toren, paard, loper of pion) en op welk speelveld hij zich bevindt. De score van een positie bestaat uit het verschil van de waarde van alle stukken van de speler aan zet en de waarde van alle stukken van de tegenpartij.

De meeste schaakprogramma's die piece-square tables implementeren maken gebruik van twee tabellen voor elk speelstuk : één voor het middenspel en één voor het eindspel. Dus de score van een stuk hangt niet alleen meer af van het type stuk en welk veld het bezet, maar ook van de fase van het spel waarin we ons bevinden. Wij kiezen bij onze implementatie voor dezelfde werkwijze.

Wij hebben verschillende piece-square tables configuraties getest en de meest effectieve behouden. Deze piece-square-tables vindt u terug in bijlage 6: piece-square tables. De waarden die we in onze tabellen gebruiken zijn gelijk aan de waarden die door Steve Maughan zijn gebruikt in zijn schaakprogramma Maverick (ELO-Rating[3]: 2.515).

[1] Realization probability search

[2] Matthew Lai: auteur van het schaakprogramma Giraffe

[3] ELO-punten: geeft de sterkte van een speler weer. De sterkste menselijke speler heeft 2853 punten en het sterkste schaakprogramma 3393 punten

7. Resultaten

7.1 Realization probability search

We hebben onze implementatie van RPS[1] geëvalueerd en gebruikten daarvoor twee belangrijke criteria:

1. De snelheid. We vergeleken de snelheid van RPS[1] met:
    - De snelheid van een eigen implementatie van een klassieke move ordering variant (piece square tables) met dezelfde testgames.
    - De snelheid van de move ordering techniek die werd gebruikt in Giraffe (4). Dit is een schaakprogramma dat speelt op grootmeester niveau (ongeveer 2.400 ELO[2] punten) en gebruik maakt van een neuraal netwerk voor de move ordering.
2. De voorspellingsgraad. We vergeleken de voorspellingsgraad van RPS[1] met de voorspellingsgraad van:
    - De geïmplementeerde klassieke move ordering (zie 2.2.3 piece square tables).
    - De move ordering techniek (neuraal netwerk) die werd gebruikt in het schaakprogramma Giraffe (4) .

De hardware waarop RPS[1] is getest is een desktop PC met een i7 3770K processor (3,5 Ghz) en 8 Gb aan ram geheugen. Als test werden twintig testgames voorgeschoteld aan het RPS[1] algoritme. Deze twintig testgames werden gespeeld tussen schaakprogramma's op topniveau (minstens een ELO[2] rating van 3.000) . Omdat het technisch nog onmogelijk is om in een schaakpositie de beste zet te bepalen - men zou immers de partij volledig moeten doorrekenen – nemen we aan dat de gespeelde zet door het schaakprogramma de beste zet is. De zet die door het RPS[1] algoritme wordt aangewezen wordt dus vergeleken met de zet die werkelijk is gespeeld door het schaakprogramma. Er werden in totaal 1.874 plies[3] onderzocht.

De beste resultaten werden behaald op drie plies[3] diep, een materiaalscore van 1 en een drempelwaarde van 0. RPS[1] is in staat om in 10,20% van alle geteste posities de beste zet te voorspellen. In 24,05% van alle geteste posities kan RPS[1] de beste zet in de top drie plaatsen, in 34,71% in de top vijf en in 53,98% in de top tien (zie figuur 20). Gemiddeld zijn er in een middenspelpositie 32 zetten mogelijk. De snelheid van dit algoritme ligt op een gemiddelde van 0,24 seconden per onderzochte positie.

7.2 Realization probability search (aangepast)

De RPS[1] score van een zet wordt berekend door het gemiddelde te nemen van de RPS[1] scores van de zetcategorieën waartoe de zet behoort. Hierdoor wordt het onderscheid tussen een goede zet en een slechte zet minder afgelijnd.

---

[1] Realization Probability Search
[2] ELO-punten: geeft de sterkte van een speler weer. De sterkste menselijke speler heeft 2853 punten en het sterkste schaakprogramma 3393 punten
[3] Ply: één zet van één speler

Daarom verdeelden we de zetcategorieën in vier groepen: de captures, de voordelige categorieën, de dringende categorieën en de overige categorieën.

In bijna alle schaakprogramma's komen in de move ordering steeds alle captures vooraan. Hiervoor komen vijf categorieën in aanmerking op onze probability tabel: we vermenigvuldigen hun probability index met vier. De dringende categorieën vereisen een dringende interactie, bijvoorbeeld: een stuk staat aangevallen en staat niet verdedigd. Het stuk moet worden verdedigd of verplaatst. Hiervoor komen op onze probability tabel ook vijf categorieën in aanmerking: we vermenigvuldigen hun probability index met drie.

De voordelige categorieën leveren steeds een strategisch, tactisch of positioneel voordeel op. We vermenigvuldigen hun probability score met twee. Hiervoor komen 24 categorieën in aanmerking.

Strategische eigenschappen van een schaakpositie gaan vooral over de veldbeheersing. Men kan bijvoorbeeld de eigen paarden ruilen met de lopers van de tegenpartij. Men heeft dan een strategisch voordeel omdat op termijn de lopers meer velden kunnen bestrijken dan de paarden. Hoe meer stukken er afgeruild worden, hoe krachtiger het loperpaar wordt. Andersom, als men de lopers afruilt tegen de paarden heeft men een strategisch nadeel.

Tactische eigenschappen zijn voor- of nadelen op korte termijn. Men heeft bijvoorbeeld een tactisch voordeel als men via een geforceerde combinatie materiaal kan winnen of de tegenstander kan mat zetten. Tactische zetten zijn bijvoorbeeld aftrekschaak, dubbele aanval, penning, … .

Positionele eigenschappen wijzen op de structuur van de schaakpositie, bijvoorbeeld een aaneenschakeling van pionnen (de pionnen verdedigen elkaar), een vrijpion (kan niet worden geslagen door een vijandelijk pion), een geïsoleerde pion (kan niet meer worden verdedigd door een eigen pion), de koning die sterk staat verdedigd, een open lijn waarlangs de zware stukken (dame en torens) kunnen manoeuvreren.

De overblijvende 90 categorieën behouden hun probability index. De factor waarmee de probability index van de gevormde groepen (captures, voordelige categorieën, dringende categorieën en overige categorieën) zijn vermenigvuldigd zijn vastgelegd door verschillende factoren uit te proberen. De gebruikte factoren leverden de beste move ordering op.

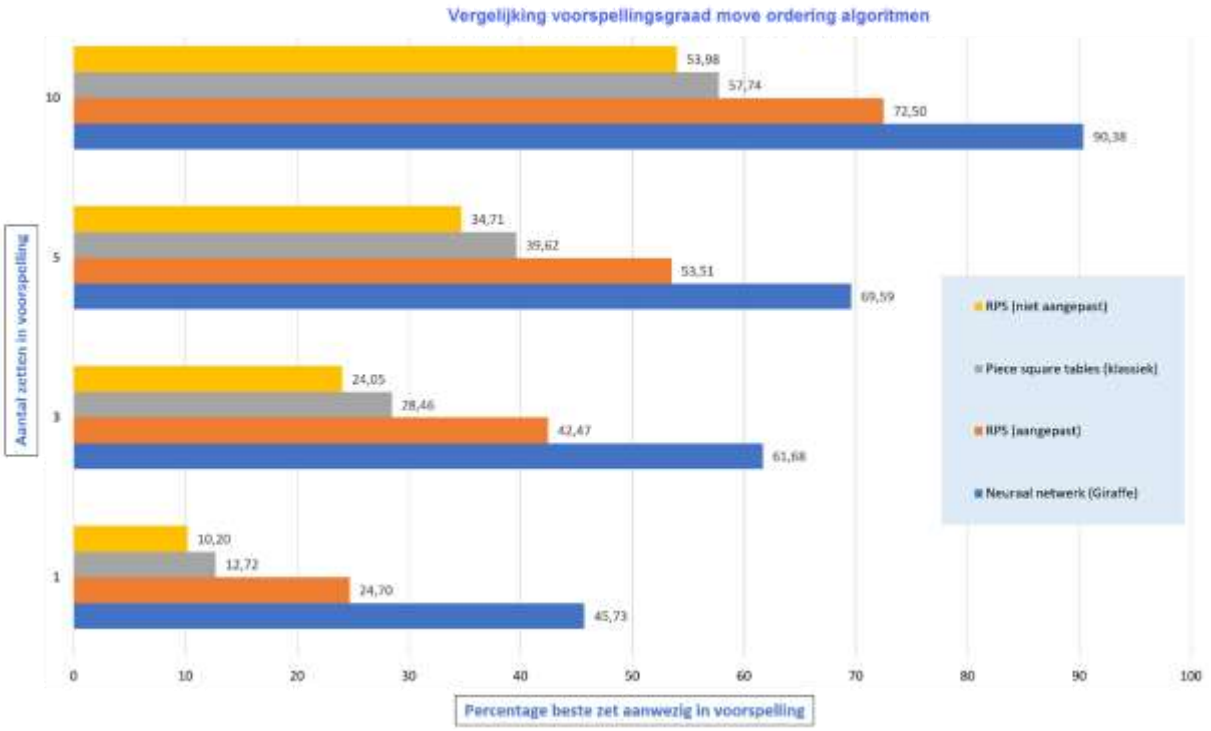
Deze aanpak had een zeer gunstig effect en zorgde ervoor dat RPS[1] in 24,7% van alle onderzochte schaakposities de beste zet kon voorspellen, in 42,47% de beste zet in de top drie kon zetten, in 53,51% de beste zet in de top 5 kon plaatsen en in 72,50% de beste zet in de top tien (zie figuur 2). Gemiddeld is er in het middenspel keuze uit 32 zetten. Er werden in totaal 1.874 plies[2] onderzocht uit twintig testmatchen die gespeeld zijn tussen top schaakprogramma's met een rating van minstens 3.000 ELO[3] punten.

Een opvallende vaststelling is dat met deze aanpak de beste resultaten werden bereikt met volgende instellingen: één ply[2] diep, een materiaalscore van -2 en een drempelwaarde van nul. Hoe dieper deze variant van RPS[1] zoekt, hoe minder accuraat de voorspellingen worden. Een voordeel is dat deze variant de resultaten ogenblikkelijk levert.

[1] Realization Probability Search

[2] Ply: één zet van één speler

[3] ELO-punten: geeft de sterkte van een speler weer. De sterkste menselijke speler heeft 2853 punten en het sterkste schaakprogramma 3393 punten



Figuur 20: Gemiddelde voorspellingsgraad: 20 testmatchen (1.874 plies[1])

7.2 Klassieke move ordering variant (piece square tables)

Onze implementatie van een typische klassieke move ordering variant (piece square tables) hebben we dezelfde twintig testgames voorgelegd als aan onze implementatie van RPS[2]. Deze klassieke variant kon in 12,72% van alle schaakposities de beste zet voorspellen. In 28,46% van alle posities kon het de beste zet in de top drie plaatsen, in 39,62% in de top vijf en in 57,74% in de top tien (zie figuur 20).

Wij hebben verschillende piece-square tables configuraties getest en de meest effectieve behouden. Deze piece-square-tables vindt u terug in bijlage 6: piece-square tables. De waarden die we in onze tabellen gebruiken zijn gelijk aan de waarden die door Steve Maughan zijn gebruikt in zijn schaakprogramma Maverick (ELO-Rating[3]: 2.515).

Ook bij deze techniek lagen de beste resultaten op één ply[1] diep. De snelheid van deze variant is nagenoeg ook ogenblikkelijk.

7.3 Neuraal netwerk

De resultaten van een move ordering algoritme op basis van een neuraal netwerk hebben we overgenomen uit de thesis van Matthew Lai, die in zijn schaakprogramma Giraffe deze techniek heeft gebruikt.

[1] Ply: één zet van één speler

[2] Realization Probability Search

[3] ELO-punten: geeft de sterkte van een speler weer. De sterkste menselijke speler heeft 2853 punten en het sterkste schaakprogramma 3393 punten

Dit move ordering algoritme kon in 45,73% van alle schaakposities de beste zet voorspellen. In 61,68% van alle posities kon het de beste zet in de top drie plaatsen, in 69,59% in de top vijf en in 90,38% in de top tien (zie figuur 20).

De snelheid van het neuraal netwerk, dat wordt gebruikt in Giraffe(4), is volgens de auteur (Matthew Lai) een negatief aspect. Het maakt de move ordering dermate traag (24) dat het zelfs op zeer krachtige machines er soms niet in slaagt een voorspelling te doen binnen de toegestane tijd en bijgevolg niet bruikbaar is als move ordering systeem voor schaakprogramma's die draaien op een normale desktop PC.

8. Conclusie

We hebben RPS[1] getest als move ordering systeem in een schaakprogramma en vergeleken met een eigen geïmplementeerde klassieke move ordering variant (piece square tables) en een move ordering systeem dat wordt gebruikt in het schaakprogramma Giraffe (4) dat gebruik maakt van een neuraal netwerk.

Uit de resultaten van een eerste implementatie konden we besluiten dat move ordering door middel van RPS[1] zonder enige aanpassingen iets slechter scoort dan de klassieke variant op basis van piece square tables.

Als we echter de zetcategorieën wijzigen in verschillende groepen en deze voorrang verlenen op elkaar, worden de resultaten plots een pak beter. De verschillende groepen bestaan uit: captures (5), dringende categorieën (5), voordelige categorieën (24) en de rest van de categorieën (90).

Deze variant scoort beduidend beter dan de klassieke variant aan dezelfde snelheid: +11,98% in de beste zet categorie, + 14,01% in de top drie, +13,89% in de top vijf en +14,76% in de top tien.

Vergeleken met het move ordering systeem dat Matthew Lai gebruikt in zijn schaakprogramma Giraffe (4) dat gebruik maakt van een neuraal netwerk, scoort de aangepaste variant van RPS[1] beduidend slechter: -21,03% in de beste zet categorie, -19,21% in de top drie, -16,08% in de top vijf en -17,88% in de top tien.

Dit is echter een moeilijke vergelijking omdat de beste zet voor Giraffe(4) de zet is die het programma uiteindelijk zal spelen. Giraffe(4) speelt momenteel op internationaal meester niveau wat betekent dat de beste zet uiteindelijk wordt bepaald door een programma van ongeveer 2.400 ELO[2] punten. We dienen op te merken dat de beste zet in een bepaalde schaakpositie momenteel nog niet kan worden bepaald – omdat een schaakprogramma dan zou moeten doorrekenen tot er een matstelling ontstaat en dat levert gewoonweg teveel varianten op om door te rekenen - en in ons onderzoek is de beste zet, de zet die momenteel door het sterkste schaakprogramma[3] zou worden gedaan in die positie.

Een tweede probleem bij het gebruik van een neuraal netwerk is de snelheid. Dit is, volgens de auteur (Matthew Lai), nog altijd een negatief aspect van neurale netwerken. Het maakt de move ordering dermate traag (24) dat het zelfs op zeer krachtige machines er soms niet in slaagt een voorspelling te doen binnen de toegestane tijd en bijgevolg niet bruikbaar is als move ordering systeem voor schaakprogramma's die draaien op een normale desktop PC.

We hebben RPS[1] met maar één klassiek move ordering systeem vergeleken. Uit deze context kunnen we concluderen dat RPS[1] waarschijnlijk goed bruikbaar is als move ordering systeem in een schaakprogramma. Veel schaakprogramma's gebruiken echter een combinatie van move ordering technieken om de resultaten te verbeteren. Toekomstige onderzoeken zullen moeten uitwijzen of RPS[1] in combinatie met andere move ordering technieken zijn voorspellend vermogen nog kan opdrijven.

[1] Realization probability search

[2] ELO-punten: geeft de sterkte van een speler weer. De sterkste menselijke speler heeft 2853 punten en het sterkste schaakprogramma 3393 punten

[3] Komodo: 3394 ELO punten

8.1 Discussie

Een mogelijkheid die we hebben overwogen is RPS[1] te implementeren in een open source schaakprogramma zodat we dan konden vergelijken met de originele move ordering techniek die in het schaakprogramma is gebruikt. Dit leek echter niet haalbaar omdat de meeste open source programma's slecht zijn gedocumenteerd en heel weinig schaakprogramma's in Java zijn geschreven (de taal waarin onze implementatie van RPS[1] is geschreven). Bovendien wordt meestal een combinatie gebruikt van move ordering technieken, zodat een eerlijke vergelijking dan weer niet opgaat. Ook qua tijdsbestek zouden we ver boven het aantal onderzoeksuren komen dat we hadden vooropgesteld.

8.2 Toekomstig werk

Bijna alle schaakprogrammeurs houden angstvallig hun move ordering algoritmen geheim en wensten daarom geen resultaten mee te delen over de technieken die zij gebruikten. Een feit is echter dat de meeste programmeurs een combinatie gebruiken van move ordering technieken om de resultaten steeds verder op te drijven. Het voorspellend vermogen van RPS[1] kan waarschijnlijk verder worden opgedreven door gebruik te maken van hulp move ordering technieken zoals bijvoorbeeld hash tables om te vermijden dat varianten die leiden tot dezelfde positie meerdere keren worden onderzocht.

Een ander onderwerp als aanvullende studie is wat het effect zou zijn als we meer of minder zetcategorieën zouden gebruiken in de probability tabel.

Momenteel zal bij het bepalen van de realization probability index van een variant deze steeds lager worden hoe dieper men zoekt, wat het hanteren van een zoekdrempel (waarde waarbij men de tak van een zoekboom snoeit) moeilijk maakt. Als men deze zoekdrempel zou kunnen normaliseren, zal waarschijnlijk het systeem nog verbeteren.

Het verklaren waarom de aangepaste variant van RPS[1] minder accuraat werkt op grotere zoekdiepte kan ook nog het onderwerp van een toekomstig onderzoek vormen.

[1] Realization probabiltiy search

Literatuurlijst

1. CCRL. (z.j.). *Rating List Chess Programs*. Geraadpleegd op 1 maart 2017: http://www.computerchess.org.uk/ccrl/4040/
2. Winands, M., & Bjornsson, Y. (2008). *Enhanced realization probability search*. New Mathematics and Natural Computation, 4(03), 329-342.
3. Yokoyama, D., Tsuruoka, Y., & Chikayama, T. (2002). *Game-tree search algorithm based on realization probability*. Icga Journal, 2002(September), 145-152.
4. Lai, M. (2015). *Giraffe: Using deep reinforcement learning to learn to play chess*. arXiv, 1509.01549 (September), 39.
5. Norvig, P., Russell, S., & Davis, E. (2010). *Artificial intelligence* (3e ed.). Harlow, Groot-Brittannië: Pearson Education, 5-6.
6. Silver, D., Huang, A., & Maddison, C. (2016). *Mastering the game of Go with deep neural networks and tree search*. Nature, 529(7587), 484-489. doi:10.1038/nature16961.
7. Shannon, C. (1950*). Programming a computer for playing chess*. Philosophical Magazine, 41(314), 256. doi:10.1080/14786445008521796.
8. FIDE. (2017, 1 maart). *World chess federation ranglijst*. Geraadpleegd op 1 maart 2017: https://ratings.fide.com/top.phtml?list=men
9. Schaeffer, J. (1989). *The History Heuristic and Alpha-Beta Search Enhancements in Practice*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 11(11), 1203-1212.
10. Winands, M., Van Der Werf, E., & Van Den Herik, H. (2006). *The relative history heuristic*. Lecture Notes in Computer Science, 3846(LNCS), 262-272. doi:10.1007/11674399_18.
11. Maughan, S. (2017, 1 maart*). Chess programming*. Geraadpleegd op 1 maart 2017: https://www.chessprogramming.net/
12. Greer, K. (2000). *Computer chess move-ordering schemes using move influence*. Artificial Intelligence, 120(2), 235-250. doi:10.1016/S0004-3702(00)00026-6.
13. Stern, D., Herbrich, R., & Graepel, T. (2006). *Bayesian pattern ranking for move prediction in the game of Go*. Paper gepresenteerd op de Proceedings of the 23rd International Conference on Machine Learning, Pittsburgh, VS. doi:10.1038/nature16961.
14. Slagle, J., & Dixon, J. (1969). *Experiments With Some Programs That Search Game Trees*. Journal of the ACM (JACM), 16(2), 189-207. doi:10.1145/321510.321511.
15. Higashiuchi, Y., & Grimbergen, R. (2006). *Enhancing Search Efficiency by Using Move Categorization Based on Game Progress in Amazons*.Advances in Computer Games: 11th International Conference, ACG 2005, Taipei, Taiwan, September 6-9, 2005. Revised Papers, 1, 73-87. doi:10.1007/11922155_6.
16. Bourzutschky, M., Tamplin, J., & Haworth, G. (2005). *Chess endgames: 6-man data and strategy*. Theoretical Computer Science, 349(2), 140-157. doi:10.1016/j.tcs.2005.09.043.
17. Lasker, E., & Mont, J. (1971). *Chess Strategy*. Oxford, Groot-Brittanië: Courier Corporation.
18. Bojkov, D., & Georgiev, V. (2015). *A Course in Chess Tactics*. London, England: Gambit publications.
19. Wilkins, D. (1980). *Using patterns and plans in chess*. Artificial Intelligence, 14(2), 165-203. doi:10.1016/0004-3702(80)90039-9.
20. Naroditsky, D. (2010). *Mastering Positional Chess*. Alkmaar, Nederland: New in chess.
21. Gelfer, I. (2010). Positional Chess Handbook: *495 Instructive Positions from Grandmaster Games,* New York, VS: Dover books.

22. Polgar, S. (2016). *Learn Chess the right way!*. Milford, USA: Russell Enterprises.
23. CCRL. (2016, 1 oktober). [Chess engines downloads]. Geraadpleegd op 1 maart 2017: http://www.computerchess.org.uk/ccrl/4040/games.html
24. MIT technologie review. Geraadpleegd op 15 maart 2017: https://www.technologyreview.com/s/541276/deep-learning-machine-teaches-itself-chess-in-72-hours-plays-at-international-master/

Bijlage 1: Afkortingen

AI: Artificial Intelligence

CCRL: Computer Chess Rating List

FIDE: Federation Internationale des Echecs

LOA: Lines of Action. Strategisch bordspel voor twee spelers.

PGN: Portable Game Notation is een vorm van schaaknotatie waarbij de gegevens van schaakpartijen worden opgeslagen in platte tekstbestanden. Het is in de jaren tachtig bedacht door een groep internetgebruikers die beoogden de uitwisselbaarheid van gegevens tussen schaaksoftware te vergroten. PGN heeft zich ontwikkeld tot een standaard. Praktisch elk schaakprogramma en elke schaakdatabase kan PGN importeren of exporteren en vele websites stellen partijenverzamelingen in PGN beschikbaar.

PV: Principal variation. Is de hoofdvariant die uiteindelijk wordt gespeeld.

RPS: Realization probability search

GM: Grootmeester

IM: Internationaal meester

Bijlage 2: Verklarende woordenlijst

Brute Force: is het gebruik van rekenkracht om een probleem op te lossen met een computer zonder gebruik te maken van algoritmen of heuristieken om de berekening te versnellen. Brute force wordt gebruikt als er geen algoritme bekend is dat sneller of efficiënter tot een oplossing leidt. De methode bestaat uit het botweg uitproberen van alle mogelijke opties, net zo lang tot er een gevonden is die overeenkomt met de gewenste invoer.

ELO: Amerikaanse natuurkundige en schaker van Hongaarse afkomst, ontwikkelde het ELO-rating systeem om spelers in te delen naar speelsterkte.

GO: Is een van origine Oost-Aziatisch bordspel voor twee spelers.

Ply: Bij een bordspel tussen twee personen is een ply één zet door één van de spelers. De term ply is geïntroduceerd omdat in verschillende tradities de begrippen zet en beurt een verschillende betekenis hebben. Bij het schaken bijvoorbeeld wordt een zet ook wel eens een halfzet genoemd; een zet van wit en een van zwart vormen dan samen één zet; bij de notatie van een partij worden in ieder geval deze combinaties van twee genummerd, niet de afzonderlijke (half)zetten.

In computerprogramma's is het begrip belangrijk, omdat een ply overeenkomt met een niveau in een beslissingsboom. Bij een schaakprogramma gaat het dan erom hoe diep wordt gezocht in de beslissingsboom van mogelijke stellingen. Hierbij wordt onderscheid gemaakt tussen de zoekdiepte (uitgedrukt in aantal ply) dat alle mogelijke stellingen worden geëvalueerd (brute force) en hoe diep selectief verder wordt gezocht naar de beste zet(ten).

Mat: Ook wel schaakmat genoemd. De koning staat mat als hij staat aangevallen en zichzelf niet aan die aanval kan onttrekken. Hij kan dus bij de volgende zet gewoon geslagen worden. De vijandelijke koning mat zetten is het doel van het spel.

Neuraal netwerk: Dit netwerk wordt opgebouwd uit verschillende (meestal zeer eenvoudige) processoren met een zeer uitgebreide onderlinge connectie. Doordat deze connecties niet standaard dezelfde blijven, maar kunnen worden verzwakt, versterkt, aangemaakt of verbroken, kan dit kunstmatig neuraal netwerk zichzelf 'trainen' en lijkt het ook veel meer op een biologisch neuraal netwerk. Er zijn bijvoorbeeld beslissingsmodellen voor de interpretatie van schommelingen in beurskoersen, op basis waarvan voorspellingen kunnen worden gedaan van toekomstige beurskoersen. Een probleem bij de toepassing van neurale netwerken in de praktijk is het feit dat het werkt via het Black Box-principe; een neuraal netwerk is niet in staat een uitleg te geven van de reden voor de gegeven uitvoer. De 'kennis' van het getrainde netwerk bestaat uit de exacte configuratie van een groot aantal coëfficiënten waaraan verder niet zoveel waar te nemen is. Men moet het model alsook simplificeren om aan de mogelijkheden van de computer tegemoet te komen, en daardoor wordt het beeld van de realiteit sterk vervormd (Wikipedia)

Positioneel voordeel: Men heeft positioneel voordeel als de positie kenmerken vertoond die als voordelig worden beschouwd zoals een aaneenschakeling van pionnen, een pion die vrij staat en dus niet meer kan geslagen worden door een vijandelijke pion, de koning die sterk staat verdedigd, een open lijn waarlangs de zware stukken kunnen manoeuvreren,…

Pruning: Snoeien. Deze term wordt gebruikt als bepaalde varianten uit de zettenboom worden verwijdert. Belangrijk is dat het snoeien geen invloed heeft op het eindresultaat.

Pat: De koning staat pat als hij niet staat aangevallen en slechts die zetten ter beschikking heeft waardoor hij wel zou aangevallen staan. In dit geval is de partij remise.

Remise: Gelijkspel. Dit kan worden overeengekomen door beide spelers of kan in sommige situaties worden afgedwongen.

Strategisch Voordeel: Men heeft een strategisch voordeel als men een betere veldbezetting kan verkrijgen bijvoorbeeld als men de paarden ruilt tegen de lopers van de tegenpartij. De lopers zullen op termijn meer velden kunnen bestrijken als paarden als er zich minder stukken op het bord bevinden.

Tactisch Voordeel: Men heeft een tactisch voordeel als men via een geforceerde combinatie materiaal kan winnen of de tegenstander kan mat zetten. De tactische combinaties worden in groepen ingedeeld zoals: aftrekschaak, dubbele aanval, penning, ….

Zoekboom: Een tree of zoekboom is een datastructuur in de informatica die een bijzonder geval van een graaf is. Hij bestaat uit een knoop(punt) of vertex (Engels: node) die de stam (ook wel wortel, Eng.: root) genoemd wordt, en die het ingangspunt is voor de in de boom opgeslagen informatie. In deze wortelknoop zitten nul of meer pointers die naar andere knooppunten verwijzen. Ieder knooppunt behalve de wortel heeft precies een ouder (Eng.: parent node) en nul of meer kinderen (Eng.: child nodes). De verwijzingen gaan dus nooit tussen de kinderen onderling maar alleen van ouder naar kind; in een wat uitgebreidere versie eventueel ook van kind naar ouder (bidirectionele graaf). In een tree bestaan geen cirkelpaden en is er altijd precies 1 pad van de wortel naar een willekeurige knoop. Een knoop die zelf geen kinderen heeft noemt men een blad (Eng.: leaf).

Bijlage 3: Zet categorieën.

Bij de beschrijving van de zetten wordt er melding gemaakt van een vertrek en aankomstveld. Het vertrekveld is het veld dat door het stuk wordt bezet voor de zet. Het aankomstveld is het veld dat door het stuk wordt bezet na de zet.

1. Mat. (tactisch)
   De zet beëindigt de partij.
2. Stuk (vertrek) geen aanvalszet. (strategisch)
   Het verplaatste stuk valt bij het vertrekveld geen vijandelijk stuk aan.
3. Geen Capture. (strategisch)
   Het verplaatste stuk slaat geen vijandig stuk.
4. Stuk (vertrek) geen verdedigingszet. (strategisch)
   Het verplaatste stuk verdedigt geen ander stuk bij het vertrekveld.
5. Stuk (aankomst) geen verdedigingszet. (strategisch)
   Het verplaatste stuk verdedigt geen ander stuk bij aankomst.
6. Stuk (aankomst) kan niet worden geslagen. (strategisch)
   Het verplaatste stuk kan niet worden geslagen door een vijandelijk stuk op het aankomstveld.
7. Stuk (vertrek) kan niet worden geslagen. (strategisch)
   Het verplaatste stuk kon niet worden geslagen door een vijandelijk stuk op het vertrekveld.
8. Zet vermeerdert mobility. (positioneel)
   Na de zet zijn er meer zetten mogelijk dan voor de zet.
9. Stuk (vertrek) kan worden geslagen door minder stuk. (strategisch)
   Het verplaatste stuk kon worden geslagen door een minder vijandelijk stuk op het vertrekveld.
10. Zet vermindert mobility. (positioneel)
    Na de zet zijn er minder zetten mogelijk dan voor de zet.
11. Stuk behoudt zicht op het centrum. (strategisch)
    Het verplaatste stuk kan het centrum nog steeds bezetten bij een volgende zet.
12. Capture hoger stuk. (tactisch)
    Een hoger vijandelijk stuk is geslagen door het verplaatst stuk.
13. Stuk (vertrek) staat verdedigd. (tactisch)
    Het verplaatste stuk stond verdedigd bij het vertrekveld.
14. Stuk (vertrek) staat meer verdedigd dan aangevallen. (tactisch)
    Het verplaatste stuk stond meer verdedigd dan aangevallen bij het vertrekveld.
15. Stuk (aankomst) staat verdedigd.(tactisch)
    Het verplaatste stuk staat verdedigd bij het aankomstveld.
16. Stuk (aankomst) geen aanvalszet. (tactisch)
    Het verplaatste stuk valt geen vijandelijk stuk aan bij het aankomstveld.
17. Stuk (aankomst) aanvalszet. (tactisch)
    Het verplaatste stuk valt een vijandelijk stuk aan bij het aankomstveld.
18. Enpassant zet. (tactisch)
    Een pion mag in de uitgangsstelling twee velden vooruit. Als het veld dat hij daarbij passeert door een pion van de tegenstander bedreigd wordt (de pion 'eindigt' dan derhalve naast de vijandelijke pion) mag die pion hem slaan ('en passant' = in het voorbijgaan), alsof hij slechts één

veld vooruit was gezet. En passant slaan is alleen toegestaan bij de zet direct volgend op de opmars van de pion.

19. Stuk (vertrek) staat onverdedigd en aangevallen. (tactisch)
    Het verplaatste stuk stond aangevallen en niet verdedigd bij het vertrekveld.
20. Stuk (aankomst) staat meer verdedigd dan aangevallen. (tactisch)
    Het verplaatste stuk staat meer verdedigd dan aangevallen bij het aankomstveld.
21. Stuk (vertrek) kan worden geslagen door gelijk stuk. (tactisch)
    Het verplaatste stuk kon worden geslagen door een gelijkwaardig stuk bij het vertrekveld.
22. Stuk (vertrek) bezet open lijn. (strategisch)
    Het verplaatste stuk bezette een open lijn (minsten 5 opeenvolgende velden vrij).
23. Dubbele toren op zevende rij. (positioneel)
    Na de zet bevinden er zich twee torens op de zevende rij.
24. Zet verhoogt velddominantie. (strategisch)
    De eigen stukken bezetten meer belangrijke velden dan in de vorige positie.
25. Stuk (vertrek) bezet rand. (strategisch)
    Het verplaatste stuk bevond zich op de rand van het bord bij het vertrekveld.
26. Drie zware stukken op één lijn ongedaan. (positioneel)
    Na de zet bevinden er zich geen drie zware stukken meer op één lijn.
27. Zet behoudt velddominantie. (strategisch)
    De eigen stukken bezetten evenveel belangrijke velden dan in de vorige positie.
28. Stuk (aankomst) staat evenveel verdedigd dan aangevallen. (tactisch)
    Het verplaatste stuk staat evenveel verdedigd dan aangevallen bij het aankomstveld.
29. Aftrekschaak. (tactisch)
    Door het stuk te verplaatsen, wordt er schaak gegeven door een ander eigen stuk.
30. Stuk behoudt zicht op het subcentrum. (strategisch)
    Het verplaatste stuk kan het subcentrum nog steeds bezetten bij een volgende zet.
31. Stuk (vertrek) staat meer aangevallen dan verdedigd. (tactisch)
    Het verplaatste stuk staat minder verdedigd dan aangevallen bij het vertrekveld.
32. Stuk (vertrek) bezet innerrand. (strategisch)
    Het verplaatste stuk stond niet op de rand of in het (sub)centrum bij het vertrekveld.
33. Stuk verliest zicht op het subcentrum. (strategisch)
    Het verplaatste stuk kan het subcentrum niet meer bezetten bij een volgende zet.
34. Capture onverdedigd stuk. (tactisch)
    Er wordt een vijandelijk stuk geslagen dat niet stond verdedigd.
35. Stuk (vertrek) kan worden geslagen. (tactisch)
    Het verplaatste stuk kan worden geslagen door een vijandelijk stuk bij het vertrekveld.
36. Capture stuk dat meer staat aangevallen dan verdedigd. (tactisch)
    Er wordt een stuk geslagen dat meer aangevallen stond dan verdedigd.
37. Stuk (vertrek) staat evenveel verdedigd dan aangevallen. (tactisch)
    Het verplaatste stuk staat evenveel verdedigd dan aangevallen bij het vertrekveld.
38. Korte Rokade. (positioneel)
    Het is de enige zet buiten de lange rokade waarbij 2 stukken van dezelfde kleur verplaatst worden: koning en toren.
39. Stuk (aankomst) bezet innerrand. (strategisch)
    Het verplaatste stuk staat niet op de rand of in het (sub)centrum bij het aankomstveld.

40. Stuk wint zicht op het subcentrum. (strategisch)
    Het verplaatste stuk kan nu het subcentrum bezetten bij een volgende zet.
41. Stuk (vertrek) aanvalszet. (tactisch)
    Het verplaatste stuk viel een vijandelijk stuk aan bij het vertrekveld.
42. Stuk is gepend (beschermt eigen koning) (tactisch)
    Verplaatst stuk heeft zich tussen aanvaller en koning verplaatst en mag zich dus niet meer verplaatsen of de eigen koning valt!
43. Stuk (aankomst) bezet subcentrum. (strategisch)
    Het verplaatste stuk bezet het subcentrum bij het aankomstveld.
44. Stuk (aankomst) verdedigingszet. (tactisch)
    Het verplaatste stuk verdedigt een ander stuk bij het aankomstveld.
45. Stuk (vertrek) bezet subcentrum. (strategisch)
    Het verplaatste stuk bezet het subcentrum bij het vertrekveld.
46. Geïsoleerde pion voor tegenstander. (positioneel)
    Een geïsoleerde pion is een pion die niet meer door andere pionnen kan worden verdedigd.
47. Stuk (vertrek) staat onverdedigd. (tactisch)
    Het verplaatste stuk stond onverdedigd bij het vertrekveld.
48. Stuk (vertrek) valt lager stuk aan. (tactisch)
    Het verplaatste stuk valt een lager stuk aan bij het vertrekveld.
49. Stuk (vertrek) verdedigingszet. (tactisch)
    Het verplaatste stuk verdedigt een ander stuk bij het vertrekveld.
50. Capture gelijk stuk. (tactisch)
    Er wordt een evenwaardig stuk geslagen.
51. Stuk (aankomst) valt lager stuk aan. (tactisch)
    Het verplaatste stuk valt een lager stuk aan bij het aankomstveld.
52. Stuk (vertrek) verdedigt lager stuk. (tactisch)
    Het verplaatste stuk verdedigt een lager stuk bij het vertrekveld.
53. Stuk is toren. (materiaal)
    Het verplaatst stuk is een toren.
54. Stuk (aankomst) bezet rand. (strategisch)
    Het verplaatste stuk bezet de rand bij het aankomstveld.
55. Pat. (tactisch)
    Pat betekent dat de speler aan beurt geen legale zet kan uitvoeren en toch niet mat staat.
56. Stuk (aankomst) verdedigt lager stuk. (tactisch)
    Het verplaatste stuk verdedigt een lager stuk bij het aankomstveld.
57. Stuk (aankomst) staat onverdedigd. (tactisch)
    Het verplaatste stuk staat onverdedigd bij het aankomstveld.
58. Geïsoleerde pion. (positioneel)
    Een geïsoleerde pion is een pion die niet meer door andere pionnen kan worden verdedigd.
59. Stuk (aankomst) kan worden geslagen. (tactisch)
    Het verplaatste stuk kan worden geslagen bij het aankomstveld.
60. Dame en toren op één verticale lijn ongedaan. (positioneel)
    Er bevinden zich na de zet geen dame en toren meer op één lijn.
61. Stuk wint zicht op het centrum. (strategisch)
    Het verplaatste stuk kan nu het centrum bezetten bij een volgende zet.

62. Stuk is paard. (materiaal)
    Het verplaatst stuk is een paard.
63. Stuk is pion. (materiaal)
    Het verplaatst stuk is een pion.
64. Zet verliest velddominantie. (strategisch)
    De eigen stukken bezetten minder belangrijke velden dan in de vorige positie.
65. Dubbelschaak. (tactisch)
    De vijandelijke koning wordt door twee stukken schaak gezet. Dubbelschaak heeft als kenmerk dat het steeds een aftrekschaak is.
66. Stuk is loper. (materiaal)
    Het verplaatst stuk is een loper.
67. Stuk is dame. (materiaal)
    Het verplaatst stuk is een dame.
68. Dubbelpion. (positioneel)
    Een dubbelpion betekent dat er twee pionnen van de eigen kleur op één verticale lijn staan.
69. Capture. (tactisch)
    Het verplaatste stuk slaat een ander stuk.
70. Stuk verliest zicht op het centrum. (strategisch)
    Het verplaatste stuk kan het centrum niet meer bezetten bij een volgende zet.
71. Verplaatsing voorpost. (positioneel)
    Een voorpost is een loper of een paard dat gedekt wordt door een pion en niet kan weggejaagd worden door vijandelijke pionnen. Bovendien bevindt het stuk zich op de helft van de tegenstander.
72. Toren verlaat zevende rij. (positioneel)
73. Zet behoudt mobility. (strategisch)
    Er kunnen evenveel zetten worden gedaan na de zet dan voor de zet.
74. Versterkt koningsstelling. (postioneel)
    Na de zet bevinden er zich meer stukken rond de koning dan voor de zet.
75. Stuk (vertrek) bezet centrum. (strategisch)
    Het verplaatste stuk bezette een centrumveld.
76. Verzwakt koningsstelling. (positioneel)
    Na de zet bevinden er zich minder stukken rond de koning dan voor de zet.
77. Stuk (aankomst) verdedigt stuk dat stond aangevallen/onverdedigd. (tactisch)
    Het verplaatste stuk verdedigt bij aankomst een stuk dat stond aangevallen en niet was verdedigd.
78. Dubbele toren op één lijn ongedaan. (positioneel)
    Na de zet bevinden er zich geen twee torens meer op één verticale lijn.
79. Stuk (aankomst) valt hoger stuk aan. (tactisch)
    Het verplaatste stuk valt bij het aankomstveld een hoger stuk aan.
80. Het verplaatst stuk veroorzaakt een pionhole. (positioneel)
    Een pionhole is een veld voor een pion waar een vijandelijke stuk kan staan, zonder dat dit stuk door een ander pion kan worden aangevallen.
81. Stuk (aankomst) kan worden geslagen door hoger stuk. (tactisch)
    Het verplaatste stuk kan bij aankomst worden geslagen door een hoger stuk.
82. Stuk is koning. (materiaal)

Het verplaatst stuk is een koning.

83. Stuk (aankomst) bezet open lijn. (positioneel)
    Het verplaatste stuk bezet bij aankomst een open lijn (minstens 5 opeenvolgende velden vrij).
84. Verwijdert triplepion. (positioneel)
    Er bevinden zich na de zet geen 3 pionnen van de eigen kleur meer op dezelfde verticale lijn.
85. Stuk (aankomst) plaatst zich tussen aanvaller en waardevoller stuk. (tactisch)
    Het verplaatst stuk staat op het aankomstveld niet absoluut gepend maar als het zich verzet, sneuvelt er een waardevoller stuk.
86. Stuk (aankomst) bezet centrum. (strategisch)
    Het verplaatst stuk bezet bij het aankomstveld een centrumveld.
87. Stuk (aankomst) valt gelijkwaardig stuk aan. (tactisch)
    Het verplaatst stuk valt bij aankomst een gelijkwaardig stuk aan.
88. Vork. (tactisch)
    Het verplaatst stuk (geen paard) valt minstens twee gelijkwaardige stukken tegelijk aan. Het verplaatst stuk kan niet worden geslagen door een minderwaardig stuk.
89. Stuk (vertrek) valt gelijkwaardig stuk aan. (tactisch)
    Het verplaatst stuk viel bij het vertrekveld een gelijkwaardig stuk aan.
90. Vijandelijke koning staat schaak. (tactisch)
    Het verplaatste stuk zet de vijandelijke koning schaak.
91. Stuk is vergevorderde pion (5de rij of meer). (positioneel)
    De verplaatste pion bevindt zich op de helft van de tegenstander.
92. Lange Rokade. (positioneel)
    Het is de enige zet buiten de korte rokade waarbij twee stukken van dezelfde kleur verplaatst worden: koning en toren. De koning bevindt zich verder van de rand dan bij korte rokade.
93. Stuk (aankomst) kan worden geslagen door gelijk stuk. (tactisch)
    Het verplaatste stuk kan bij aankomst worden geslagen door een gelijkwaardig stuk.
94. Vrije pion. (postioneel)
    Een vrije pion is een pion die niet door vijandelijke pionnen kan worden gestopt.
95. Triplepion. (positioneel)
    Er bevinden zich na de zet 3 pionnen van de eigen kleur op één verticale lijn.
96. Vrije Pion Tegenstander. (positioneel)
    Een vrije pion is een pion die niet door vijandelijke pionnen kan worden gestopt.
97. Het verplaatste stuk vormt een voorpost. (positioneel)
    Een voorpost is een loper of een paard dat gedekt wordt door een pion en niet kan weggejaagd worden door vijandelijke pionnen. Bovendien bevindt het stuk zich op de helft van de tegenstander.
98. Stuk (aankomst) verdedigt gelijkwaardig stuk. (tactisch)
    Het verplaatste stuk verdedigt bij aankomst een gelijkwaardig stuk.
99. Stuk (aankomst) staat minder verdedigd dan aangevallen. (tactisch)
    Het verplaatste stuk staat bij aankomst minder verdedigd dan aangevallen.
100. Verwijdert dubbelpion. (postioneel)
     Na de zet is er een dubbelpion minder dan voor de zet.
101. Capture lager stuk. (tactisch)
     Het verplaatst stuk slaat een lager stuk.
102. Dubbele toren op één lijn. (positoneel)

Er bevinden zich na de zet twee eigen torens op één verticale lijn.

103. Paardvork. (tactisch)
Een paard dat minstens twee gelijkwaardige stukken aanvalt.
104. Pion promoveert tot dame. (tactisch)
105. Capture stuk dat evenveel staat aangevallen als verdedigd. (tactisch)
Het verplaatste stuk slaat een stuk dat evenveel staat aangevallen dan verdedigd.
106. Stuk (vertrek) is randpion. (strategisch)
Het verplaatste stuk is een pion dat zich op de rand van het bord bevond.
107. Loper (aankomst) bezet lange diagonaal. (positioneel)
Het verplaatste stuk is een loper die bij aankomst een lange diagonaal bezet.
108. Dame en toren op één lijn. (positioneel)
Na de zet bevinden er zich een dame en toren van de eigen kleur op één verticale lijn.
109. Stuk (aankomst) is randpion. (strategisch)
Het verplaatste stuk is een pion en bezet de rand bij aankomst.
110. Loper (vertrek) verlaat lange diagonaal. (positioneel)
Het verplaatste stuk is een loper die de lange diagonaal verlaat.
111. Vijandelijk stuk gepend (voor vijandelijke koning). (tactisch)
Het verplaatste stuk pent een vijandelijk stuk absoluut. Het vijandelijk stuk mag zich meer verzetten of zijn koning valt.
112. Capture stuk dat minder staat aangevallen dan verdedigd. (tactisch)
Het verplaatst stuk slaat een stuk dat minder stond aangevallen dan verdedigd.
113. Vijandelijk stuk gepend (voor vijandelijk stuk). (tactisch)
Het vijandelijk stuk wordt niet absoluut gepend maar als het vijandelijk stuk zich verzet, dan sneuvelt er een waardevoller stuk.
114. Stuk (vertrek) verdedigt gelijkwaardig stuk. (tactisch)
Het verplaatst stuk verdedigt een gelijkwaardig stuk bij het vertrekveld.
115. Toren op zevende rij. (positioneel)
Het verplaatst stuk is een toren en bezet de zevende rij.
116. Stuk (vertrek) verwijdert zich tussen aanvaller en waardevoller stuk. (tactisch)
Als gevolg hiervan kan de tegenstander een hoger stuk slaan.
117. Stuk (aankomst) verdedigt hoger stuk. (tactisch)
Het verplaatst stuk verdedigt een hoger stuk bij aankomst.
118. Stuk (vertrek) kan worden geslagen door hoger stuk. (tactisch)
Het verplaatst stuk kon worden geslagen door een hoger stuk op het vertrekveld.
119. Drie zware stukken op één lijn. (positioneel)
Na de zet bevinden er zich 3 zware stukken op één verticale lijn.
120. Stuk (vertrek) verdedigt hoger stuk. (tactisch)
Het verplaatst stut verdedigde een hoger stuk bij het vertrekveld.
121. Stuk (aankomst) kan worden geslagen door lager stuk. (tactisch)
Het verplaatst stuk kan bij aankomst worden geslagen door een lager stuk.
122. Stuk (aankomst) staat onverdedigd en aangevallen. (tactisch)
Het verplaatst stuk staat bij aankomst onverdedigd en aangevallen.
123. Stuk (vertrek) valt hoger stuk aan. (tactisch)
Het verplaatst stuk viel bij vertrek een hoger stuk aan.
Pion promoveert tot ander stuk dan dame.

Bijlage 4: Handleiding software

analyseCategorienDetail.jar

Dit Java programma analyseert de partijen in het pgn bestand en geeft voor elke positie die voorkomt in de partij alle mogelijke categorieën (zie probability tabel) die kunnen worden gespeeld en de categorie die effectief is gespeeld in die positie.

Het programma geeft ook een samenvatting van hoeveel keer de categorieën (zie probabilitytabel) werkelijk zijn gespeeld in alle posities die voorkomen in de partijen en hoeveel keer ze mogelijks konden worden gespeeld.

Dit programma neemt als argument een pgn bestand. Het programma wordt als volgt uitgevoerd:

java -jar AnalyseCategorienDetails.jar *.pgn

Er wordt ook in de directory waar het programma runt een log folder aangemaakt, die een logbestand bevat waarin alle uitvoer is opgenomen.

Vb uitvoer met pgn bestand dat één partij bevat*:*

Er wordt een log bewaard in de log folder (wordt aangemaakt in map waar programma runt).

We testen of het pgnbestand correcte partijen bevat…
1: Komodo 10.1 64-bit - Hannibal 1.7 64-bit, CCRL 40/40 (524.1.901) , CCRL 0 [A45]
Einde van bestand bereikt!
Aantal correcte partijen: 1

Onderzoek partij: 1: Komodo 10.1 64-bit - Hannibal 1.7 64-bit, CCRL 40/40 (524.1.901) , CCRL 0 [A45]

Ply 1: Mogelijke categorie: Verplaatst stuk (vertrek) staat verdedigd - Na3
Ply 1: Mogelijke categorie: Verplaatst stuk (Aankomst) staat verdedigd bij aankomst - Na3
Ply 1: Mogelijke categorie: Verplaatst stuk (vertrek) staat meer verdedigd dan aangevallen - Na3
Ply 1: Mogelijke categorie: Verplaatst stuk (Aankomst) staat meer verdedigd dan aangevallen - Na3
Ply 1: Mogelijke categorie: stuk behoudt dominantie: Na3
Ply 1: Mogelijke categorie: Zicht op centrum ongewijzigd - Na3
Ply 1: Mogelijke categorie: Zicht op subcentrum ongewijzigd - Na3
Ply 1: Mogelijke categorie: Stuk (aankomst) geen aanvalszet - Na3
Ply 1: Mogelijke categorie: Stuk (vertrek) geen aanvalszet - Na3
Ply 1: Mogelijke categorie: Stuk (aankomst) Geen verdedigingszet - Na3
Ply 1: Mogelijke categorie: Stuk (vertrek) Geen verdedigingszet - Na3
Ply 1: Mogelijke categorie: Geen capture - Na3
Ply 1: Mogelijke categorie: Paardzet - Na3
Ply 1: Stuk (aankomst) kan niet worden geslagen - Na3

Ply 1: Stuk (vertrek) kan niet worden geslagen - Na3
Ply 1: Stuk (aankomst) bezet rand - Na3
Ply 1: Stuk (vertrek) bezet rand - Na3
Ply 1: stuk verplaatst dat stond aangevallen en onverdedigd - Na3
Ply 1: Mogelijke categorie: behoudt mobility - Na3
Ply 1: Mogelijke categorie: stuk wint dominantie: Nc3
Ply 1: Mogelijke categorie: stuk wint zicht op het centrum: Nc3
Ply 1: Mogelijke categorie: Stuk verliest zicht op subcentrum: Nc3
Ply 1: stuk (aankomst) bezet subcentrum - Nc3
Ply 1: Mogelijke categorie: vermeerdert mobility - Nc3
Ply 1: Mogelijke categorie: Randpion (aankomst) - a3
Ply 1: Mogelijke categorie: Randpion (vertrek) - a3
Ply 1: Mogelijke categorie: Pionzet - a3
Ply 1: Mogelijke categorie: vermindert mobility - a3
Ply 1: Mogelijke categorie: Veroorzaakt pionhole: - b3
Ply 1: stuk (aankomst) bezet innerrand - b3
Ply 1: stuk (vertrek) bezet innerrand - b3
Ply 1: Mogelijke categorie: Verplaatst stuk (Aankomst) staat evenveel verdedigd dan aangevallen - b4
Ply 1: Mogelijke categorie: Stuk staat onverdedigd - b4
Ply 1: Mogelijke categorie: Stuk verliest zicht op centrum: c3
Ply 1: Mogelijke categorie: Verzwakt koningsstelling - d3
Ply 1: Verplaatst stuk (vertrek) staat verdedigd - d4
Ply 1: Verplaatst stuk (Aankomst) staat verdedigd bij aankomst - d4
Ply 1: Verplaatst stuk (vertrek) staat meer verdedigd dan aangevallen - d4
Ply 1: Verplaatst stuk (Aankomst) staat meer verdedigd dan aangevallen - d4
Ply 1: Stuk wint dominantie: d4
Ply 1: Zicht op centrum ongewijzigd - d4
Ply 1: Stuk verliest zicht op subcentrum: d4
Ply 1: Stuk (aankomst) geen aanvalszet - d4
Ply 1: Stuk (vertrek) geen aanvalszet - d4
Ply 1: Stuk (aankomst) Geen verdedigingszet - d4
Ply 1: Stuk (vertrek) Geen verdedigingszet - d4
Ply 1: Geen capture - d4
Ply 1: Verzwakt koningsstelling - d4
Ply 1: Pionzet - d4
Ply 1: Mogelijke categorie: stuk (aankomst) bezet centrum - d4
Ply 1: stuk (aankomst) bezet centrum - d4
Ply 1: stuk (vertrek) bezet innerrand - d4
Ply 1: Stuk (aankomst) kan niet worden geslagen - d4
Ply 1: Stuk (vertrek) kan niet worden geslagen - d4
Ply 1: stuk verplaatst dat stond aangevallen en onverdedigd - d4
Ply 1: vermeerdert mobility - d4
Ply 2: Mogelijke categorie: Verplaatst stuk (vertrek) staat verdedigd - Na6
Ply 2: Mogelijke categorie: Verplaatst stuk (Aankomst) staat verdedigd bij aankomst - Na6
Ply 2: Mogelijke categorie: Verplaatst stuk (vertrek) staat meer verdedigd dan aangevallen - Na6

Ply 2: Mogelijke categorie: Verplaatst stuk (Aankomst) staat meer verdedigd dan aangevallen - Na6
Ply 2: Mogelijke categorie: stuk behoudt dominantie: Na6
Ply 2: Mogelijke categorie: Zicht op centrum ongewijzigd - Na6
Ply 2: Mogelijke categorie: Zicht op subcentrum ongewijzigd - Na6
Ply 2: Mogelijke categorie: Stuk (aankomst) geen aanvalszet - Na6
Ply 2: Mogelijke categorie: Stuk (vertrek) geen aanvalszet - Na6
Ply 2: Mogelijke categorie: Stuk (aankomst) Geen verdedigingszet - Na6
Ply 2: Mogelijke categorie: Stuk (vertrek) Geen verdedigingszet - Na6
Ply 2: Mogelijke categorie: Geen capture - Na6
Ply 2: Mogelijke categorie: Paardzet - Na6
Ply 2: Stuk (aankomst) kan niet worden geslagen - Na6
Ply 2: Stuk (vertrek) kan niet worden geslagen - Na6
Ply 2: Stuk (aankomst) bezet rand - Na6
Ply 2: Stuk (vertrek) bezet rand - Na6
Ply 2: stuk verplaatst dat stond aangevallen en onverdedigd - Na6
Ply 2: Mogelijke categorie: behoudt mobility - Na6
Ply 2: Mogelijke categorie: stuk wint dominantie: Nc6
Ply 2: Mogelijke categorie: stuk wint zicht op het centrum: Nc6
Ply 2: Mogelijke categorie: Stuk verliest zicht op subcentrum: Nc6
Ply 2: Mogelijke categorie: Stuk (aankomst) valt lager stuk aan - Nc6
Ply 2: Mogelijke categorie: Stuk (aankomst) is aanvalszet - Nc6
Ply 2: stuk (aankomst) bezet subcentrum - Nc6
Ply 2: Mogelijke categorie: vermeerdert mobility - Nc6
Ply 2: Verplaatst stuk (vertrek) staat verdedigd - Nf6
Ply 2: Verplaatst stuk (Aankomst) staat verdedigd bij aankomst - Nf6
Ply 2: Verplaatst stuk (vertrek) staat meer verdedigd dan aangevallen - Nf6
Ply 2: Verplaatst stuk (Aankomst) staat meer verdedigd dan aangevallen - Nf6
Ply 2: Stuk wint dominantie: Nf6
Ply 2: Stuk wint zicht op het centrum: Nf6
Ply 2: Stuk verliest zicht op subcentrum: Nf6
Ply 2: Stuk (aankomst) geen aanvalszet - Nf6
Ply 2: Stuk (vertrek) geen aanvalszet - Nf6
Ply 2: Stuk (aankomst) Geen verdedigingszet - Nf6
Ply 2: Stuk (vertrek) Geen verdedigingszet - Nf6
Ply 2: Geen capture - Nf6
Ply 2: Paardzet - Nf6
Ply 2: stuk (aankomst) bezet subcentrum - Nf6
Ply 2: Stuk (aankomst) kan niet worden geslagen - Nf6
Ply 2: Stuk (vertrek) kan niet worden geslagen - Nf6
Ply 2: Stuk (vertrek) bezet rand - Nf6
Ply 2: stuk verplaatst dat stond aangevallen en onverdedigd - Nf6
Ply 2: vermeerdert mobility - Nf6
Ply 2: Mogelijke categorie: Verplaatst stuk (Aankomst) staat evenveel verdedigd dan aangevallen - Nh6
Ply 2: Stuk (aankomst) kan worden geslagen door gelijkwaardig stuk - Nh6

Ply 2: Stuk (aankomst) kan worden geslagen - Nh6
Ply 2: Mogelijke categorie: Randpion (aankomst) - a6
Ply 2: Mogelijke categorie: Randpion (vertrek) - a6
Ply 2: Mogelijke categorie: Pionzet - a6
Ply 2: Mogelijke categorie: vermindert mobility - a6
Ply 2: Mogelijke categorie: Veroorzaakt pionhole: - b6
Ply 2: stuk (aankomst) bezet innerrand - b6
Ply 2: stuk (vertrek) bezet innerrand - b6
Ply 2: Mogelijke categorie: Stuk staat onverdedigd - b5
Ply 2: Mogelijke categorie: Stuk verliest zicht op centrum: c6
Ply 2: Mogelijke categorie: Verplaatst stuk (Aankomst) staat meer aangevallen dan verdedigd - c5
Ply 2: Mogelijke categorie: Stuk (aankomst) valt gelijkwaardig stuk aan - c5
Ply 2: Mogelijke categorie: Stuk staat onverdedigd en aangevallen - c5
Ply 2: Mogelijke categorie: Verzwakt koningsstelling - d6
Ply 2: stuk (aankomst) bezet centrum - d5
Ply 2: Stuk (aankomst) kan worden geslagen door hoger stuk - g5
…

Samenvatting: Gespeelde categorieën in partijen

Aantal onderzochte partijen: 1
Aantal onderzochte plies: 60
Aantal onderzochte mogelijke plies: 1957

Lange Rokade: 0/8 - Probability Index = ,00%
Korte Rokade: 1/23 - Probability Index = 4,35%
Captures gelijk stuk: 4/27 - Probability Index = 14,81%
Captures lager stuk: 5/39 - Probability Index = 12,82%
Captures hoger stuk: 4/4 - Probability Index = 100,00%
Vijandelijke koning staat schaak: 4/25 - Probability Index = 16,00%
Pion promoveert tot dame: 0/0 - Probability Index = ?%
Pion promoveert tot ander stuk dan dame: 0/0 - Probability Index = ?%
zet is enpassant zet: 0/0 - Probability Index = ?%
Stuk behoudt dominantie: 23/58 - Probability Index = 39,66%
Stuk verliest dominantie: 12/51 - Probability Index = 23,53%
Stuk wint dominantie: 25/59 - Probability Index = 42,37%
Stuk verliest zicht op het centrum: 14/56 - Probability Index = 25,00%
Stuk wint zicht op het centrum: 13/55 - Probability Index = 23,64%
Stuk behoudt zicht op het centrum: 33/60 - Probability Index = 55,00%
Stuk verliest zicht op het subcentrum: 19/57 - Probability Index = 33,33%
Stuk wint zicht op het subcentrum: 14/54 - Probability Index = 25,93%
Stuk behoudt zicht op het subcentrum: 27/60 - Probability Index = 45,00%
…

analyseCategorienSamenvatting.jar

Dit Java programma analyseert de partijen in het pgn bestand en geeft een samenvatting van hoeveel keer de categorieën (zie probabilitytabel) werkelijk zijn gespeeld in alle posities die voorkomen in de partijen en hoeveel keer ze mogelijks konden worden gespeeld.

Dit programma neemt als argument een pgn bestand. Het programma wordt als volgt uitgevoerd:

java -jar AnalyseCategorienSamenvatting.jar *.pgn

Er wordt ook in de directory waar het programma runt een log folder aangemaakt, die een logbestand bevat waarin alle uitvoer is opgenomen.

Vb uitvoer met pgn bestand dat één partij bevat:

Er wordt een log bewaard in de log folder (wordt aangemaakt in map waar programma runt).

Einde van bestand bereikt!
Aantal correcte partijen: 1
Onderzoek partij: 1: Komodo 10.1 64-bit - Hannibal 1.7 64-bit, CCRL 40/40 (524.1.901) , CCRL 0 [A45]

Samenvatting: Gespeelde categorien in partijen

Aantal onderzochte partijen: 1
Aantal onderzochte plies: 60
Aantal onderzochte mogelijke plies: 1957

Lange Rokade: 0/8 - Probability Index = ,00%
Korte Rokade: 1/23 - Probability Index = 4,35%
Captures gelijk stuk: 4/27 - Probability Index = 14,81%
Captures lager stuk: 5/39 - Probability Index = 12,82%
Captures hoger stuk: 4/4 - Probability Index = 100,00%
Vijandelijke koning staat schaak: 4/25 - Probability Index = 16,00%
Pion promoveert tot dame: 0/0 - Probability Index = ?%
Pion promoveert tot ander stuk dan dame: 0/0 - Probability Index = ?%
zet is enpassant zet: 0/0 - Probability Index = ?%
Stuk behoudt dominantie: 23/58 - Probability Index = 39,66%
Stuk verliest dominantie: 12/51 - Probability Index = 23,53%
Stuk wint dominantie: 25/59 - Probability Index = 42,37%
Stuk verliest zicht op het centrum: 14/56 - Probability Index = 25,00%
Stuk wint zicht op het centrum: 13/55 - Probability Index = 23,64%
Stuk behoudt zicht op het centrum: 33/60 - Probability Index = 55,00%
Stuk verliest zicht op het subcentrum: 19/57 - Probability Index = 33,33%
Stuk wint zicht op het subcentrum: 14/54 - Probability Index = 25,93%

Stuk behoudt zicht op het subcentrum: 27/60 - Probability Index = 45,00%
Vork (Geen paard): 0/0 - Probability Index = ?%
PaardVork: 0/0 - Probability Index = ?%
Capture onverdedigd stuk: 7/16 - Probability Index = 43,75%
Capture stuk dat evenveel staat aangevallen als verdedigd: 3/30 - Probability Index = 10,00%
Capture stuk dat meer staat aangevallen dan verdedigd: 9/25 - Probability Index = 36,00%
Capture stuk dat meer staat verdedigd dan aangevallen: 1/23 - Probability Index = 4,35%
Verplaatst stuk (vertrek) staat meer verdedigd dan aangevallen: 32/56 - Probability Index = 57,14%
Verplaatst stuk (vertrek) staat evenveel verdedigd dan aangevallen: 14/55 - Probability Index = 25,45%
Verplaatst stuk (vertrek) staat meer aangevallen dan verdedigd: 10/26 - Probability Index = 38,46%
Verplaatst stuk (aankomst) staat meer verdedigd dan aangevallen: 23/56 - Probability Index = 41,07%
Verplaatst stuk (aankomst) staat evenveel verdedigd dan aangevallen: 24/56 - Probability Index = 42,86%
Verplaatst stuk (aankomst) staat meer aangevallen dan verdedigd: 9/58 - Probability Index = 15,52%
Dubbele toren op één lijn: 1/1 - Probability Index = 100,00%
Dame en toren op één lijn: 0/11 - Probability Index = ,00%
3 zware stukken op één lijn: 0/1 - Probability Index = ,00%
DubbelPion: 2/24 - Probability Index = 8,33%
TriplePion: 0/0 - Probability Index = ?%
Stuk (vertrek) is randpion: 1/56 - Probability Index = 1,79%
Stuk (aankomst) is randpion: 1/56 - Probability Index = 1,79%
Geisoleerde pion: 1/12 - Probability Index = 8,33%
Geisoleerde pion voor tegenstander: 1/10 - Probability Index = 10,00%
Vrije Pion: 2/11 - Probability Index = 18,18%
Vrije Pion Tegenstander: 2/10 - Probability Index = 20,00%
Stuk (aankomst) bezet Open Lijn: 2/14 - Probability Index = 14,29%
Stuk (vertrek) bezet Open Lijn: 2/9 - Probability Index = 22,22%
AftrekSchaak: 0/0 - Probability Index = ?%
DubbelSchaak: 0/0 - Probability Index = ?%
Stuk is gepend (beschermt koning): 2/3 - Probability Index = 66,67%
Vijandelijk stuk gepend (voor vijandelijke koning): 0/2 - Probability Index = ,00%
Vijandelijk stuk gepend (voor vijandelijk stuk): 1/15 - Probability Index = 6,67%
Stuk (aankomst) plaatst zich tussen aanvaller en waardevoller stuk: 0/0 - Probability Index = ?%
Stuk (Vertrek) verwijdert zich tussen aanvaller en waardevoller stuk: 0/1 - Probability Index = ,00%
Veroorzaakt pionhole: 7/56 - Probability Index = 12,50%
Stuk (aankomst) is een voorpost: 1/7 - Probability Index = 14,29%
Stuk (vertrek) is een voorpost: 2/2 - Probability Index = 100,00%
Geeft Pat: 0/0 - Probability Index = ?%
Stuk (aankomst) valt lager stuk aan: 16/56 - Probability Index = 28,57%
Stuk (aankomst) valt gelijkwaardig stuk aan: 7/55 - Probability Index = 12,73%
Stuk (aankomst) valt hoger stuk aan: 12/40 - Probability Index = 30,00%
Stuk (vertrek) valt lager stuk aan: 6/23 - Probability Index = 26,09%
Stuk (vertrek) valt gelijkwaardig stuk aan: 1/8 - Probability Index = 12,50%
Stuk (vertrek) valt hoger stuk aan: 0/1 - Probability Index = ,00%

Stuk (aankomst) is geen verdedigingszet: 46/60 - Probability Index = 76,67%
Stuk (aankomst) verdedigt lager stuk: 9/48 - Probability Index = 18,75%
Stuk (aankomst) verdedigt gelijkwaardig stuk: 4/12 - Probability Index = 33,33%
Stuk (aankomst) verdedigt hoger stuk: 1/13 - Probability Index = 7,69%
Stuk (vertrek) is geen verdedigingszet: 45/60 - Probability Index = 75,00%
Stuk (vertrek) verdedigt lager stuk: 12/31 - Probability Index = 38,71%
Stuk (vertrek) verdedigt gelijkwaardig stuk: 2/28 - Probability Index = 7,14%
Stuk (vertrek) verdedigt hoger stuk: 1/8 - Probability Index = 12,50%
Stuk staat onverdedigd: 9/59 - Probability Index = 15,25%
Stuk (aankomst) staat onverdedigd en aangevallen: 0/56 - Probability Index = ,00%
Geen capture: 47/60 - Probability Index = 78,33%
Dubbele toren op zevende rij: 0/0 - Probability Index = ?%
PionZet: 16/57 - Probability Index = 28,07%
DameZet: 11/48 - Probability Index = 22,92%
TorenZet: 12/45 - Probability Index = 26,67%
LoperZet: 8/53 - Probability Index = 15,09%
PaardZet: 9/48 - Probability Index = 18,75%
KoningsZet: 4/57 - Probability Index = 7,02%
Stuk (aankomst) kan worden geslagen: 21/58 - Probability Index = 36,21%
Stuk (aankomst) kan worden geslagen door minder stuk: 4/53 - Probability Index = 7,55%
Stuk (aankomst) kan worden geslagen door gelijkwaardig stuk: 5/54 - Probability Index = 9,26%
Stuk (aankomst) kan worden geslagen door hoger stuk: 13/57 - Probability Index = 22,81%
Stuk (aankomst) kan niet worden geslagen: 39/60 - Probability Index = 65,00%
Stuk (vertrek) kan worden geslagen: 20/43 - Probability Index = 46,51%
Stuk (vertrek) kan worden geslagen door minder stuk: 7/8 - Probability Index = 87,50%
Stuk (vertrek) kan worden geslagen door gelijkwaardig stuk: 4/17 - Probability Index = 23,53%
Stuk (vertrek) kan worden geslagen door hoger stuk: 9/137 - Probability Index = 6,57%
Stuk (vertrek) kan niet worden geslagen: 40/60 - Probability Index = 66,67%
Stuk (aankomst) bezet rand: 21/60 - Probability Index = 35,00%
Stuk (vertrek) bezet rand: 26/58 - Probability Index = 44,83%
Stuk verplaatst dat stond aangevallen en onverdedigd: 28/56 - Probability Index = 50,00%
Stuk slaat een ander stuk: 13/44 - Probability Index = 29,55%
Stuk staat bij aankomst verdedigd: 37/60 - Probability Index = 61,67%
Stuk staat bij vertrek verdedigd: 39/60 - Probability Index = 65,00%
Stuk (vertrek) aanvalszet: 7/25 - Probability Index = 28,00%
Stuk (aankomst) aanvalszet: 35/58 - Probability Index = 60,34%
Stuk (vertrek) verdedigingszet: 15/41 - Probability Index = 36,59%
Stuk (aankomst) verdedigingszet: 14/49 - Probability Index = 28,57%
Stuk staat bij vertrek onverdedigd: 12/48 - Probability Index = 25,00%
Behoudt mobility: 5/55 - Probability Index = 9,09%
Vermindert mobility: 15/56 - Probability Index = 26,79%
Vermeerdert mobility: 40/60 - Probability Index = 66,67%
Stuk (aankomst) is geen aanvalszet: 25/60 - Probability Index = 41,67%
Stuk (vertrek) is geen aanvalszet: 53/60 - Probability Index = 88,33%
Stuk (aankomst) valt lager stuk aan: 16/56 - Probability Index = 28,57%

Stuk (aankomst) valt gelijkwaardig stuk aan: 7/55 - Probability Index = 12,73%
Stuk (aankomst) valt hoger stuk aan: 12/40 - Probability Index = 30,00%
Stuk (vertrek) valt lager stuk aan: 6/23 - Probability Index = 26,09%
Stuk (vertrek) valt gelijkwaardig stuk aan: 1/8 - Probability Index = 12,50%
Stuk (vertrek) valt hoger stuk aan: 0/1 - Probability Index = ,00%
Dubbele toren op één lijn: 1/1 - Probability Index = 100,00%
Dame en toren op één lijn: 0/11 - Probability Index = ,00%
3 zware stukken op één lijn: 0/1 - Probability Index = ,00%
Dubbele toren op één lijn ongedaan: 0/1 - Probability Index = ,00%
Dame en toren op één lijn ongedaan: 0/0 - Probability Index = ?%
3 zware stukken op één lijn ongedaan: 0/0 - Probability Index = ?%
Loper (vertrek) verlaat lange diagonaal: 0/23 - Probability Index = ,00%
Loper (aankomst) bezet lange diagonaal: 2/24 - Probability Index = 8,33%
Verzwakt koningsstelling: 12/53 - Probability Index = 22,64%
Versterkt koningsstelling: 9/57 - Probability Index = 15,79%
DubbelPion ongedaan: 1/9 - Probability Index = 11,11%
TriplePion ongedaan: 0/0 - Probability Index = ?%
DubbelPion: 2/24 - Probability Index = 8,33%
TriplePion: 0/0 - Probability Index = ?%
Stuk is vergevorderde pion: 1/5 - Probability Index = 20,00%
Toren op zevende rij: 0/4 - Probability Index = ,00%
Toren verlaat zevende rij: 0/0 - Probability Index = ?%
Stuk (aankomst) bezet centrum: 4/55 - Probability Index = 7,27%
Stuk (aankomst) bezet subcentrum: 16/57 - Probability Index = 28,07%
Stuk (aankomst) bezet innerrand: 18/60 - Probability Index = 30,00%
Stuk (vertrek) bezet centrum: 2/28 - Probability Index = 7,14%
Stuk (vertrek) bezet subcentrum: 7/52 - Probability Index = 13,46%
Stuk bezet (vertrek) innerrand: 24/60 - Probability Index = 40,00%
Tijd gespendeerd: 0 seconden

RPS.jar / RPS_nietAangepast.jar

Deze Java programma's analyseren de partijen in een pgn bestand en geven per positie in een partij volgende informatie weer:

- **Positie** in de partij.
- **Aantal gegenereerde reeksen**: mogelijke varianten op aangegeven zoekdiepte in die positie.
- **Aantal gefilterde reeksen (RPS)**: varianten waarmee geen rekening werd gehouden omdat ze een RPS score hebben lager dan de opgegeven drempelwaarde.
- **Aantal gefilterde reeksen (materiaal)**: varianten waarmee geen rekening werd gehouden omdat ze een materiaal score hebben lager dan de opgegeven minimum materiaalscore.
- **Berekende beste zet**: de beste variant die werd berekend (ranking 1) met daarbij de RPS score en de materiaalscore na het spelen van die variant.

- **Werkelijk gespeelde zet in de partij**: de werkelijk gespeelde variant in de partij met daarbij de RPS score en de materiaalscore na het spelen van die variant.

Het verschil tussen RPS.jar en RPS_nietAangepast.jar is dat bij RPS.jar voor de berekening van de RPS score de categorieën zijn opgesplitst in vier groepen en dat deze groepen voorrang krijgen op elkaar (zie hoofdstuk Realization probability search)

Na het bereken van alle zetten in een partij wordt een rapport opgemaakt hoeveel keer de berekende beste werkelijk de beste zet was en hoeveel keer de beste zet in de top 30 voor kwam. Vervolgens wordt ook de gemiddelde rekentijd per positie aangegeven.

Deze programma's worden als volgt uitgevoerd:

Java -jar (RPS.jar op RPS_nietAangepast.jar) **zoekdiepte startmove endmove minDrempelwaarde minMateriaalscore**

- **Zoekdiepte**: geeft aan hoeveel plies diep er moet worden gezocht.
- **Startmove**: geeft aan bij welke ply moet worden gestart met move ordering.
- **Endmove**: geeft aan bij welke ply[1] er moet worden gestopt met move ordering. (Voor alle zetten geef je een 0 in)
- **Drempelwaarde**: geeft de cutoff[1] waarde (tussen 0 en 1) aan. Alle zetten met een probability lager dan de cutoff[1] waarde wordt dan geen rekening mee gehouden. Dit kan als gevolg hebben, dat met een goede zet die in eerste instantie niet goed werd beoordeeld geen rekening wordt gehouden. Indien je geen cutoff[1] waarde wilt opgeven en rekening wilt houden met alle zetten, dien je 0 op te geven.
- **minMateriaalScore**: geeft het gewenste niveau van materiaalwinst aan op de gewenste zoekdiepte. 0 betekent dat er geen materiaalwinst is voor beide partijen.

Een voorbeeld:

java -jar RPS.jar TestGame1.pgn 3 0 0 0.47 -2

In dit geval wordt voor alle posities in de partij voor alle mogelijke zetten in die positie drie plies[2] diep gerekend. Bij een probability index van minder dan 0.47 wordt met die zet geen rekening meer gehouden. Er mag maximaal een verlies zijn van 2 eenheden (pionnen) bij elke eindpositie.

Uitvoer ziet er als volgt uit:

Er wordt een log bewaard in de log folder (wordt aangemaakt in de map logs in de directory waar programma runt).

Partij ingelezen: Komodo 10.1 64-bit - Hannibal 1.7 64-bit, CCRL 40/40 (523.1.902) , CCRL 0 [B12] - Aantal zetten: 46

[1] Cutoff: snoeien van een tak in de zoekboom

[2] Ply: één zet van één speler

Einde van bestand bereikt!

Er bevinden zich 1 partij(en) in TestGame1.pgn

Partij 1: Komodo 10.1 64-bit - Hannibal 1.7 64-bit, CCRL 40/40 (523.1.902) , CCRL 0 [B12]

Starting moveordering van plyNr: 0 tot 91

….

Move ordering voor ply 35: Raxc1

Aantal gegenereerde reeksen: 28757
Aantal gefilterde reeksen (RPS waarde minder dan 0.47): 24483
Aantal gefilterde reeksen (MateriaalScore minder dan -2.0): 2689
Aantal overblijvende reeksen op 3 levels diep: 1585

Ranking: 1: Bg5 (0.81)-Rxa1 (1.4)-Rxa1 (1.24)-
RPS Score: 1.41
Materiaal Score: -2.0

Gespeelde zet in partij:
Ranking: 6: Raxc1 (1.22)-Nxe5 (0.64)-Ndxe5 (1.26)-
RPS Score: 0.98
Materiaal Score: 5.0

Move ordering voor ply 36: Kxd7

Aantal gegenereerde reeksen: 30416
Aantal gefilterde reeksen (RPS waarde minder dan 0.47): 5742
Aantal gefilterde reeksen (MateriaalScore minder dan -2.0): 24529
Aantal overblijvende reeksen op 3 levels diep: 145

Ranking: 1: Nxh2 (0.72)-Nxf8 (0.98)-Rxf8 (1.1)-
RPS Score: 0.78
Materiaal Score: -2.0

Gespeelde zet in partij:
Ranking: 7: Kxd7 (0.75)-Nd4 (0.71)-Nxe5 (1.08)-
RPS Score: 0.58
Materiaal Score: 1.0

Move ordering voor ply 37: Rfd1+

Aantal gegenereerde reeksen: 26311
Aantal gefilterde reeksen (RPS waarde minder dan 0.47): 25861
Aantal gefilterde reeksen (MateriaalScore minder dan -2.0): 270
Aantal overblijvende reeksen op 3 levels diep: 180

Ranking: 1: Nd4 (0.71)-c5 (0.76)-Nxf5 (1.44)-
RPS Score: 0.78
Materiaal Score: 3.0

Move ordering voor ply 38: Kc7

Aantal gegenereerde reeksen: 4815
Aantal gefilterde reeksen (RPS waarde minder dan 0.47): 4727
Aantal gefilterde reeksen (MateriaalScore minder dan -2.0): 38
Aantal overblijvende reeksen op 3 levels diep: 50

Gespeelde zet in partij:
Ranking: 1: Kc7 (0.81)-Nd4 (0.65)-Nxe5 (1.08)-
RPS Score: 0.57
Materiaal Score: 1.0

Move ordering voor ply 39: h3

Aantal gegenereerde reeksen: 34083
Aantal gefilterde reeksen (RPS waarde minder dan 0.47): 33584
Aantal gefilterde reeksen (MateriaalScore minder dan -2.0): 361
Aantal overblijvende reeksen op 3 levels diep: 138

Ranking: 1: Rc4 (0.74)-Ne3 (0.71)-Rxc6+ (1.24)-
RPS Score: 0.65
Materiaal Score: 1.0

Gespeelde zet in partij:
Ranking: 9: h3 (0.56)-Nxe5 (0.93)-Nxe5 (1.08)-
RPS Score: 0.56
Materiaal Score: 2.0

…

Rapport voor partij 1: Komodo 10.1 64-bit - Hannibal 1.7 64-bit, CCRL 40/40 (523.1.902) , CCRL 0 [B12]
==========

Aantal keren beste zet voorspeld op plaats 1 in onze top 15 beste zetten: 15
Aantal keren beste zet voorspeld op plaats 2 in onze top 15 beste zetten: 7
Aantal keren beste zet voorspeld op plaats 3 in onze top 15 beste zetten: 10
Aantal keren beste zet voorspeld op plaats 4 in onze top 15 beste zetten: 4
Aantal keren beste zet voorspeld op plaats 5 in onze top 15 beste zetten: 2
Aantal keren beste zet voorspeld op plaats 6 in onze top 15 beste zetten: 5
Aantal keren beste zet voorspeld op plaats 7 in onze top 15 beste zetten: 5
Aantal keren beste zet voorspeld op plaats 8 in onze top 15 beste zetten: 5
Aantal keren beste zet voorspeld op plaats 9 in onze top 15 beste zetten: 3
Aantal keren beste zet voorspeld op plaats 10 in onze top 15 beste zetten: 2
Aantal keren beste zet voorspeld op plaats 11 in onze top 15 beste zetten: 2

Aantal keren beste zet voorspeld op plaats 12 in onze top 15 beste zetten: 1
Aantal keren beste zet voorspeld op plaats 13 in onze top 15 beste zetten: 1
Aantal keren beste zet voorspeld op plaats 14 in onze top 15 beste zetten: 2
Aantal keren beste zet voorspeld op plaats 15 in onze top 15 beste zetten: 2
Aantal keren beste zet voorspeld op plaats 16 in onze top 15 beste zetten: 0
Aantal keren beste zet voorspeld op plaats 17 in onze top 15 beste zetten: 1
Aantal keren beste zet voorspeld op plaats 18 in onze top 15 beste zetten: 0
Aantal keren beste zet voorspeld op plaats 19 in onze top 15 beste zetten: 0
Aantal keren beste zet voorspeld op plaats 20 in onze top 15 beste zetten: 1
Aantal keren beste zet voorspeld op plaats 21 in onze top 15 beste zetten: 1
Aantal keren beste zet voorspeld op plaats 22 in onze top 15 beste zetten: 1
Aantal keren beste zet voorspeld op plaats 23 in onze top 15 beste zetten: 0
Aantal keren beste zet voorspeld op plaats 24 in onze top 15 beste zetten: 0
Aantal keren beste zet voorspeld op plaats 25 in onze top 15 beste zetten: 0
Aantal keren beste zet voorspeld op plaats 26 in onze top 15 beste zetten: 0
Aantal keren beste zet voorspeld op plaats 27 in onze top 15 beste zetten: 0
Aantal keren beste zet voorspeld op plaats 28 in onze top 15 beste zetten: 0
Aantal keren beste zet voorspeld op plaats 29 in onze top 15 beste zetten: 1
Aantal keren beste zet voorspeld op plaats 30 in onze top 15 beste zetten: 0

Aantal plies in partij: 91 op zoekdiepte: 3

Aantal keer beste zet voorspeld: 15 op 91 plies
Percentage beste zetvoorspelling: 16.48%

Aantal keer beste zet in onze top 3: 32 op 91 plies
Percentage beste zet in onze top 3: 35.16%

Aantal keer beste zet in onze top 5: 38 op 91 plies
Percentage beste zet in onze top 5: 41.76%

Aantal keer beste zet in onze top 10: 58 op 91 plies
Percentage beste zet in onze top 10: 63.74%

Aantal keer beste zet in onze top 15: 64 op 91 plies
Percentage beste zet in onze top 15: 70.33%

Aantal keer beste zet buiten onze top 15: 27 op 91 plies
Percentage beste zet buiten onze top 15: 29.67%

Verstreken tijd: 131.0 seconden
Gemiddelde tijd per zet: 1.44 seconden

KlassiekMoveOrdering.jar

Deze Java programma's analyseren de partijen in een pgn bestand en geven per positie in een partij volgende informatie weer:

- **Positie** in de partij.

- **Aantal dubbele reeksen**: varianten waarbij geen rekening wordt gehouden omdat ze tot éénzelfde positie leiden.
- **Aantal unieke reeksen**: gegenereerde varianten min de dubbele reeksen
- **Berekende beste zet**: de beste variant die werd berekend (ranking 1) met daarbij de materiaalscore na het spelen van die variant.
- **Werkelijk gespeelde zet in de partij**: de werkelijk gespeelde variant in de partij met daarbij de de materiaalscore na het spelen van die variant.

Dit programma wordt als volgt uitgevoerd:

Java -jar KlassiekMoveOrdering.jar **zoekdiepte startmove endmove**

Zoekdiepte: geeft aan hoeveel plies diep er moet worden gezocht.
Startmove: geeft aan bij welke ply[1] moet worden gestart met move ordering.
Endmove: geeft aan bij welke ply[1] er moet worden gestopt met move ordering. (Voor alle zetten geef je een hoge waarde in bv 500)

java -jar KlassiekMoveOrdering.jar TestGame1.pgn 3 0 500

In dit geval wordt voor alle posities in de partij voor alle mogelijke zetten in die positie drie plies[1] diep gerekend.

Uitvoer ziet er als volgt uit:

Er wordt een log bewaard in de log folder (wordt aangemaakt in de map logs in de directory waar programma runt).

Partij ingelezen: Komodo 10.1 64-bit - Hannibal 1.7 64-bit, CCRL 40/40 (523.1.902) , CCRL 0 [B12] - Aantal zetten: 46

Einde van bestand bereikt!

Er bevinden zich 1 partij(en) in TestGame1.pgn

Starting moveordering van zetnr: 0 tot 46

Partij 1: Komodo 10.1 64-bit - Hannibal 1.7 64-bit, CCRL 40/40 (523.1.902) , CCRL 0 [B12]

….

Move ordering voor zet 43: Nd4

Aantal dubbele reeksen gedetecteerd: 17796
Aantal unieke reeksen gegenereerd op 3 levels diep: 25920

Ranking: 1 Bf6-Bh7-Rxd5- Score: 173.0

[1] Ply: één zet van één speler

Gespeelde zet in partij:
Ranking: 4: Nd4-Bh7-Nxe6+- Score: 160.0

Move ordering voor zet 44: Bd3

Aantal dubbele reeksen gedetecteerd: 16304
Aantal unieke reeksen gegenereerd op 3 levels diep:21373

Ranking: 1 Ne3-Nc2-Bb4- Score: 44.0

Gespeelde zet in partij:
Ranking: 7 Bd3-Nc2-Bb4- Score: 19.0

Move ordering voor zet 45: Rd1

Aantal dubbele reeksen gedetecteerd: 20064
Aantal unieke reeksen gegenereerd op 3 levels diep:30753

Ranking: 1 Rxd5-Bh7-Nxe6+- Score: 197.0

Gespeelde zet in partij:
Ranking: 7 Rd1-Bh7-Nxe6+- Score: 165.0

…

Aantal keren beste zet voorspeld op plaats 1 in onze top 15 beste zetten: 8
Aantal keren beste zet voorspeld op plaats 2 in onze top 15 beste zetten: 6
Aantal keren beste zet voorspeld op plaats 3 in onze top 15 beste zetten: 2
Aantal keren beste zet voorspeld op plaats 4 in onze top 15 beste zetten: 4
Aantal keren beste zet voorspeld op plaats 5 in onze top 15 beste zetten: 3
Aantal keren beste zet voorspeld op plaats 6 in onze top 15 beste zetten: 7
Aantal keren beste zet voorspeld op plaats 7 in onze top 15 beste zetten: 6
Aantal keren beste zet voorspeld op plaats 8 in onze top 15 beste zetten: 3
Aantal keren beste zet voorspeld op plaats 9 in onze top 15 beste zetten: 4
Aantal keren beste zet voorspeld op plaats 10 in onze top 15 beste zetten: 2
Aantal keren beste zet voorspeld op plaats 11 in onze top 15 beste zetten: 4
Aantal keren beste zet voorspeld op plaats 12 in onze top 15 beste zetten: 5
Aantal keren beste zet voorspeld op plaats 13 in onze top 15 beste zetten: 3
Aantal keren beste zet voorspeld op plaats 14 in onze top 15 beste zetten: 0
Aantal keren beste zet voorspeld op plaats 15 in onze top 15 beste zetten: 0

Rapport voor partij 1: Komodo 10.1 64-bit - Hannibal 1.7 64-bit, CCRL 40/40 (523.1.902) , CCRL 0 [B12]
==========

Aantal zetten in partij: 91 op zoekdiepte: 3

Aantal keer beste zet voorspeld: 8 op 91 zetten
Percentage beste zetvoorspelling: 8.79%

Aantal keer beste zet in onze top 3: 16 op 91 zetten
Percentage beste zet in onze top 3: 17.58%

Aantal keer beste zet in onze top 5: 23 op 91 zetten
Percentage beste zet in onze top 5: 25.27%

Aantal keer beste zet in onze top 10: 45 op 91 zetten
Percentage beste zet in onze top 10: 49.45%

Aantal keer beste zet in onze top 15: 57 op 91 zetten
Percentage beste zet in onze top 15: 62.64%

Aantal keer beste zet buiten onze top 15: 34 op 91 zetten
Percentage beste zet buiten onze top 15: 37.36%

Verstreken tijd: 3.0 seconden
Gemiddelde tijd per zet: 0.03 seconden

Bijlage 5: Probability Tabel

Probability Index Tabel Komodo (3382 ELO)-rest vd wereld (>3050 ELO)

Aantal onderzochte partijen: 3346
Aantal onderzochte posities: 418.480
Aantal onderzochte mogelijke zetten: 12.659.891

ID	Categorieën	Probability Index %
1	Mat	100,00
2	Stuk (vertrek) geen aanvalszet	89,21
3	Geen Capture	83,91
4	Stuk (vertrek) geen verdedigingszet	81,43
5	Stuk (aankomst) geen verdedigingszet	78,44
6	Stuk (aankomst) kan niet worden geslagen	76,87
7	Stuk (vertrek) kan niet worden geslagen	76,44
8	Zet vermeerdert mobility	76,28
9	Stuk (vertrek) kan worden geslagen door minder stuk	70,64
10	Zet vermindert mobility	63,85
11	Stuk behoudt zicht op het centrum	62,27
12	Capture hoger stuk	59,08
13	Stuk (vertrek) staat verdedigd	57,73
14	Stuk (vertrek) staat meer verdedigd dan aangevallen	52,84
15	Stuk (aankomst) staat verdedigd	52,60
16	Stuk (aankomst) geen aanvalszet	52,20
17	Stuk (aankomst) aanvalszet	49,82
18	Enpassant zet	49,04
19	Stuk (vertrek) staat onverdedigd en aangevallen	47,98
20	Stuk (aankomst) staat meer verdedigd dan aangevallen	44,10
21	Stuk (vertrek) kan worden geslagen door gelijk stuk	43,96
22	Stuk (vertrek) bezet open lijn	43,89
23	Dubbele toren op zevende rij	42,94
24	Zet verhoogt velddominantie	41,93
25	Stuk (vertrek) bezet rand	40,42
26	3 zware stukken op één lijn ongedaan	39,40
27	Zet behoudt velddominantie	38,60
28	Stuk (aankomst) staat evenveel verdedigd dan aangevallen	38,34
29	AftrekSchaak	38,21

ID	Categorieën	Probability Index %
30	Stuk behoudt zicht op het subcentrum	37,97
31	Stuk (vertrek) staat meer aangevallen dan verdedigd	35,56
32	Stuk (vertrek) bezet innerrand	35,18
33	Stuk verliest zicht op het subcentrum	34,85
34	Capture onverdedigd stuk	34,05
35	Stuk (vertrek) kan worden geslagen	34,03
36	Capture stuk dat meer staat aangevallen dan verdedigd	33,18
37	Stuk (vertrek) staat evenveel verdedigd dan aangevallen	32,33
38	Korte Rokade	31,99
39	Stuk (aankomst) bezet inner rand	31,31
40	Stuk wint zicht op het subcentrum	30,71
41	Stuk (vertrek) aanvalszet	28,66
42	Stuk is gepend (beschermt eigen koning)	28,46
43	Stuk (aankomst) bezet subcentrum	28,42
44	Stuk (aankomst) verdedigingszet	28,42
45	Stuk (vertrek) bezet subcentrum	28,42
46	Geisoleerde pion voor tegenstander	27,93
47	Stuk (vertrek) staat onverdedigd	27,21
48	Stuk (vertrek) valt lager stuk aan	27,15
49	Stuk (vertrek) verdedigingszet	27,07
50	Capture gelijk stuk	26,90
51	Stuk (aankomst) valt lager stuk aan	26,83
52	Stuk (vertrek) verdedigt lager stuk	26,37
53	Stuk is toren	26,10
54	Stuk (aankomst) bezet rand	25,80
55	Pat	25,00
56	Stuk (aankomst) verdedigt lager stuk	24,80
57	Stuk (aankomst) staat onverdedigd	24,33
58	Geisoleerde pion	24,23

ID	Categorieën	Probability Index %
59	Stuk (aankomst) kan worden geslagen	24,21
60	Dame en toren op één lijn ongedaan	23,78
61	Stuk wint zicht op het centrum	23,54
62	Stuk is paard	23,48
63	Stuk is pion	23,38
64	Zet verliest velddominantie	23,13
65	DubbelSchaak	22,99
66	Stuk is loper	22,80
67	Stuk is dame	22,12
68	DubbelPion	21,81
69	Capture	21,08
70	Stuk verliest zicht op het centrum	20,63
71	Verplaatsing voorpost	20,31
72	Toren verlaat zevende rij	20,20
73	Zet behoudt mobility	19,74
74	Versterkt koningsstelling	19,50
75	Stuk (vertrek) bezet centrum	18,56
76	Verzwakt koningsstelling	16,93
77	Stuk (aankomst) verdedigt stuk dat stond aangevallen/onverdedigd	16,28
78	Dubbele toren op één lijn ongedaan	16,08
79	Stuk (aankomst) valt hoger stuk aan	15,74
80	Veroorzaakt pionhole	15,63
81	Stuk (aankomst) kan worden geslagen door hoger stuk	15,19
82	Stuk is koning	14,91
83	Stuk (aankomst) bezet Open Lijn	14,16
84	Verwijdert triplepion	13,68
85	Stuk (aankomst) plaatst zich tussen aanvaller en waardevoller stuk	13,52
86	Stuk (aankomst) bezet centrum	13,44
87	Stuk (aankomst) valt gelijkwaardig stuk aan	12,02
88	Vork	11,90
89	Stuk (vertrek) valt gelijkwaardig stuk aan	11,81
90	Vijandelijke koning staat schaak	11,55
91	Stuk is vergevorderde pion (5de rij of meer)	11,42

ID	Categorieën	Probability Index %
92	Lange Rokade	11,27
93	Stuk (aankomst) kan worden geslagen door gelijk stuk	11,27
94	Vrije pion	11,26
95	TriplePion	10,61
96	Vrije Pion Tegenstander	10,56
97	Voorpost	9,79
98	Stuk (aankomst) verdedigt gelijkwaardig stuk	9,58
99	Stuk (aankomst) staat minder verdedigd dan aangevallen	9,53
100	Verwijdert DubbelPion	9,50
101	Capture lager stuk	9,35
102	Dubbele toren op één lijn	8,18
103	PaardVork	8,06
104	Pion promoveert tot dame	7,95
105	Capture stuk dat evenveel staat aangevallen als verdedigd	7,24
106	Stuk (vertrek) is randPion	7,11
107	Stuk (aankomst) bezet lange diagonaal	6,83
108	Dame en toren op één lijn	6,67
109	Stuk (aankomst) is randPion	6,53
110	Loper (vertrek) verlaat lange diagonaal	6,01
111	Vijandelijk stuk gepend (voor vijandelijke koning)	5,96
112	Capture stuk dat minder staat aangevallen dan verdedigd	5,77
113	Vijandelijk stuk gepend (voor vijandelijk stuk)	5,64
114	Stuk (vertrek) verdedigt gelijkwaardig stuk	5,46
115	Toren op zevende rij	5,26
116	Stuk (vertrek) verwijdert zich zich tussen aanvaller en waardevoller stuk	5,08
117	Stuk (aankomst) verdedigt hoger stuk	4,94
118	Stuk (vertrek) kan worden geslagen door hoger stuk	4,91
119	3 zware stukken op één lijn	4,53
120	Stuk (vertrek) verdedigt hoger stuk	4,12
121	Stuk (aankomst) kan worden geslagen door lager stuk	2,95
122	Stuk (aankomst) staat onverdedigd en aangevallen	1,52
123	Stuk (vertrek) valt hoger stuk aan	0,84
124	Pion promoveert tot ander stuk dan dame	0,75

Bijlage 6: Piece-square tables

WitPionMiddenveld = { 0, 0, 0, 0, 0, 0, 0, 0, -21, -16, -6, -1, -1, -6, -16, -21, -21, -16,-6, 4, 4, -6, -16, -21, -21, -16, -1, 9, 9, -1, -16, -21, -14, -8, 6, 17, 17, 6, -8, -14, -5, 1, 14, 29, 29, 14, 1, -5, 7, 11, 23, 39, 39, 23, 11, 7, 0, 0, 0, 0, 0, 0, 0, 0 };

WitPionEindSpel = { 0, 0, 0, 0, 0, 0, 0, 0, 5, -10, -20, -25, -25, -20, -10, 5, 5, -10,-20, -25, -25, -20, -10, 5, 10, -5, -15, -20, -20, -15, -5, 10, 18, 2, -8, -15, -15, -8, 2, 18, 30, 14, 1, -10, -10, 1, 14, 30, 45, 30, 16, 5, 5, 16, 30, 45, 0, 0, 0, 0, 0, 0, 0, 0 };

WitPaardMiddenspel = { -69, -19, -24, -14, -14, -24, -19, -69, -54, -39, -9, 11, 11, -9,-39, -54, -39, 1, 31, 21, 21, 31, 1, -39, -39, 11, 41, 36, 36, 41, 11, -39, -39, 41, 51, 51, 51, 51, 41, -39, -39, 46, 61, 71, 71, 61, 46, -39, -39, 21, 41, 41, 41, 41, 21, -39, -59, -39, -29, -29, -29, -29, -39,-59 };

WitPaardEindspel = { -63, -53, -43, -43, -43, -43, -53, -63, -53, -43, 18, 28, 28, 18, -43,-53, -43, 18, 48, 38, 38, 48, 18, -43, -43, 38, 58, 68, 68, 58, 38, -43, -43, 38, 73, 78, 78, 73, 38, -43,-43, 28, 78, 73, 73, 78, 28, -43, -53, -43, 38, 48, 48, 38, -43, -53, -63, -53, -43, -43, -43, -43, -53,-63 };

WitLoperMiddenspel = { -30, -25, -20, -20, -20, -20, -25, -30, -28, 11, 6, 1, 1, 6, 11,-28, -25, 6, 16, 11, 11, 16, 6, -25, 1, 1, 16, 21, 21, 16, 1, 1, 1, 21, 21, 26, 26, 21, 21, 1, 1, 11, 21, 26, 26, 21, 11, 1, -10, 11, 1, 1, 1, 1, 11, -10, -20, -18, -16, -14, -14, -16, -18, -20 };

WitLoperEindspel = { -38, -18, -8, 2, 2, -8, -18, -38, -18, -8, 2, 7, 7, 2, -8, -18, -8, 2,10, 12, 12, 10, 2, -8, 2, 12, 16, 20, 20, 16, 12, 2, 2, 12, 17, 22, 22, 17, 12, 2, -8, 2, 20, 22, 22, 20, 2,-8, -18, -8, 0, 12, 12, 0, -8, -18, -38, -18, -8, 2, 2, -8, -18, -38 };

WitTorenMiddenspel = { -8, -6, 2, 7, 7, 2, -6, -8, -8, -6, 2, 7, 7, 2, -6, -8, -8, -6, 6,7, 7, 6, -6, -8, -8, -6, 6, 7, 7, 6, -6, -8, -8, -6, 6, 8, 8, 6, -6, -8, -8, -6, 6, 10, 10, 6, -6, -8, 2, 2, 7, 12, 12, 7, 2, 2, -8, -6, 2, 7, 7, 2, -6, -8 };

WitTorenEindspel = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,0, 0, 0, 0, 0 };

WitDameMiddenspel = { -26, -16, -6, 4, 4, -6, -16, -26, -16, -11, -1, 4, 4, -1, -11, -16,-6, -6, -1, 4, 4, -1, -6, -6, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4 };

WitDameEindspel = { -46, -41, -31, -26, -26, -31, -41, -46, -31, -26, -16, -6, -6, -16, -26, -31, -16, -1, 14, 24, 24, 14, -1, -16, -6, 9, 24, 34, 34, 24, 9, -6, -6, 9, 24, 34, 34, 24, 9, -6, -6, 9, 24, 34, 34, 24, 9, -6, -16, 4, 19, 29, 29, 19, 4, -16, -26, -6, -1, 4, 4, -1, -6, -26 };

WitKoningMiddenspel = { -20, 0, 0, -10, -10, 0, 0, -20, -30, -30, -30, -35, -35, -30, -30, -30, -40, -40, -45, -50, -50, -45, -40, -40, -50, -50, -55, -60, -60, -55, -50, -50, -55, -55, -60, -70,-70, -60, -55, -55, -55, -55, -60, -70, -70, -60, -55, -55, -55, -55, -60, -70, -70, -60, -55, -55, -55,-55, -60, -70, -70, -60, -55, -55 };

WitKoningEindspel = { -30, -25, -15, -10, -10, -15, -25, -30, -15, -10, 0, 10, 10, 0, -10,-15, 0, 15, 30, 40, 40, 30, 15, 0, 10, 25, 40, 50, 50, 40, 25, 10, 10, 25, 40, 50, 50, 40, 25, 10, 0, 20, 35, 45, 45, 35, 20, 0, -10, 10, 15, 20, 20, 15, 10, -10 };

Bijlage 7: Resultaten voor alle geïmplementeerde move ordering algoritmen.

Piece square table					
Top	1	3	5	10	15
testgame1	13,19	29,67	42,86	54,95	70,33
testgame2	15,18	33,04	43,75	58,04	65,18
testgame3	10,92	31,09	46,22	60,50	80,67
testgame4	9,02	22,56	32,33	50,38	66,92
testgame5	10,75	23,66	31,18	52,69	69,82
testgame6	8,86	25,32	39,87	60,76	70,25
testgame7	10,95	25,55	37,96	54,01	72,99
testgame8	9,09	21,21	36,36	61,36	75,00
testgame9	19,10	32,58	38,20	51,69	67,42
testgame10	12,00	26,67	34,67	52,00	66,67
testgame11	7,07	22,22	29,29	47,47	68,69
testgame12	20,00	36,84	48,42	65,26	70,53
testgame13	8,16	26,53	44,90	57,14	69,39
testgame14	18,84	33,33	47,83	69,57	88,41
testgame15	12,12	27,27	36,36	60,61	66,67
testgame16	15,66	36,14	42,67	61,45	75,90
testgame17	3,57	16,67	30,95	54,76	65,48
testgame18	18,75	32,50	42,50	66,25	81,25
testgame19	14,77	31,82	44,32	61,36	72,73
testgame20	16,36	34,55	41,82	54,55	72,73
Percentage	12,72	28,46	39,62	57,74	71,85

RPS					
Top	1	3	5	10	15
testgame1	28,57	45,05	62,54	76,92	85,71
testgame2	23,21	43,75	58,93	75,00	79,46
testgame3	20,17	42,02	52,94	78,99	88,24
testgame4	23,31	40,60	54,14	66,17	80,45
testgame5	27,96	43,01	53,76	69,89	81,72
testgame6	20,89	39,87	50,00	70,89	82,28
testgame7	21,17	35,77	45,26	70,07	81,75
testgame8	21,97	42,42	56,06	72,73	83,33
testgame9	24,72	44,94	56,18	74,16	84,27
testgame10	26,67	44,00	60,00	73,33	85,33
testgame11	18,18	38,38	55,56	74,75	87,88
testgame12	25,26	37,89	49,47	69,47	82,11
testgame13	30,61	46,94	51,02	69,39	79,59
testgame14	33,33	50,72	60,87	78,26	85,51
testgame15	27,27	39,39	45,45	66,67	69,70
testgame16	26,51	53,01	59,04	75,90	81,93
testgame17	23,81	40,48	44,05	57,14	70,24
testgame18	21,25	38,75	48,75	76,25	85,00
testgame19	27,27	44,32	53,41	79,55	87,50
testgame20	21,82	38,18	52,73	74,55	78,18
Percentage	24,70	42,47	53,51	72,50	82,01

Niet aangepast RPS					
Top	1	3	5	10	15
testgame1	6,59	27,47	40,66	62,64	74,73
testgame2	12,50	26,79	41,96	58,04	71,43
testgame3	10,08	28,57	42,02	63,03	76,47
testgame4	10,53	21,80	33,08	47,37	56,39
testgame5	5,38	16,13	31,18	53,76	64,52
testgame6	9,49	26,58	36,71	54,43	66,46
testgame7	8,03	15,33	24,09	48,18	67,88
testgame8	9,09	23,48	32,58	57,58	71,21
testgame9	6,74	23,60	33,71	53,93	67,42
testgame10	9,33	29,33	37,33	53,33	62,67
testgame11	8,08	23,23	28,28	48,48	62,63
testgame12	13,68	24,21	34,74	49,47	58,95
testgame13	4,08	14,29	20,41	38,78	51,02
testgame14	18,48	33,33	43,48	59,42	73,91
testgame15	9,09	21,21	48,48	57,58	60,61
testgame16	14,46	22,89	30,12	50,60	65,06
testgame17	8,33	17,86	23,81	48,81	63,10
testgame18	13,75	31,25	42,50	66,25	71,25
testgame19	13,64	22,73	27,27	53,41	62,50
testgame20	12,73	30,91	41,82	54,55	63,64
percentage	10,20	24,05	34,71	53,98	65,59